uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Ajith Kamath
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

User-Credential Based Role Mapping in Multi-Domain Collaborative Environments
TITRE DE LA THÈSE / TITLE OF THESIS

Ramiro Liscano
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

A. El Saddik

T. White

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

User-Credential Based Role Mapping in Multi-Domain Collaborative Environments

By

Ajith Kamath, B.E.

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of

the requirements for the degree of

Master of Computer Science

Under the auspices of the

Ottawa-Carleton Institute for Computer Science

University of Ottawa

Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 978-0-494-34076-9
Our file Notre référence
ISBN: 978-0-494-34076-9

I hereby declare that I am the sole author of the thesis.

I authorize the University of Ottawa to lend the thesis to other institutions or individuals for the purpose of scholarly research.

Ajith Kamath

I authorize the University of Ottawa to reproduce the thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Ajith Kamath

Abstract

Collaboration between multiple organizations creates new opportunities for businesses. With such collaborations becoming a reality, it is necessary to have an access control policy integration approach to form a global policy consistent with the partner organizations. Research on policy integration has led to the proposal of several frameworks to uniformly express policies and to integrate such policies. But most of these frameworks are complex and compromise the privacy of the constituent domains by sharing all the components of an access control policy including access control lists.

In this thesis, a unique policy integration technique is described to merge Role-Based Access Control (RBAC) policies of multiple-security domains in a heterogeneous environment. The proposed mechanism uses user credentials associated with roles as the main criteria in mapping inter-domain roles. Integration of the proposed policy greatly minimizes the administration overhead while efficiently merging the policies in a heterogeneous environment. Then, an approach to extend the community-based authorization framework to include the proposed integration tool is presented. A practical implementation is provided that enables collaboration among autonomous domains.

Keywords

Policy Integration, Role-Based Access Control (RBAC), Community Authorization Service (CAS)

Acknowledgments

First and foremost, I am very thankful to my supervisor Dr. Ramiro Liscano for his invaluable guidance, encouragement, constructive criticism, and patience throughout the course of this work. It is my honor and privilege to work with a true gentleman like him.

I would like to thank Dr. Abdulmotaleb El Saddik for his valuable comments and for giving me the opportunity to be a part of Multimedia Communication Research (MCR) Lab.

I would also like to thank everyone in the project "Web service feasibility for sensor networks" for their useful comments and patience.

I would like to thank the financial supports from Instantel Inc., CITO, and the University of Ottawa.

Special thanks to Mr. Suresha Nayak for his support and help throughout my master's program.

Lastly, and most importantly, I wish to thank my mother, Muktha L. Kamath and my brothers, Arvind Kamath and Ananth Kamath, who supported me and loved me.

Ajith Kamath

Table of Contents

List of Abbreviations

Abbreviation or Acronym	Meaning
ACL	Access Control List
CA	Certification Authority
CAS	Community Authorization Service
GSI	Grid Security Infrastructure
PEP	Policy Enforcement Point
PDP	Policy Decision Point
PKI	Public Key Infrastructure
RBAC	Role Based Access Control
SAML	Security Assertion Markup Language
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSO	Single Sign-On
TTP	Trusted Third Party
URI	Uniform Resource Identifier
VO	Virtual Organization
XACML	eXtensible Access Cotnrol Markable Language
XML	eXtensible Markable Language
XPRAS	XML Permission to Role Assignment Sheet
XPS	XML Permission Sheet
XRS	XML Role Sheet
XURAS	XML User Role Assignment Sheet
XUS	XML User Sheet

List of Tables

List of Figures

Chapter 1 Introduction and Outline

The operating environment of organizations has changed drastically during the last decade. Increased connectivity has facilitated distributed multiple organization interoperation. However, with this increase in information sharing, there is a growing concern of security in such a diverse environment. Different organizations forming independent domains not only aim for the sharing of resources but also for secure access to them. The following research project investigates security issues associated with a heterogeneous environment involving interoperation between multiple organizations. This chapter introduces the motivating scenario, a description of the problem, thesis contributions, and the structure of this thesis.

1.1 Motivating Scenario

Like many heterogeneous operating environments, sensor network applications may involve the use of a variety of services from different domains. For example, a sensor network application to find the weather conditions at a given region in the city may use services – such as a sensory data service (to read the temperature, humidity, vibration etc.), a GPS service, and a mapping service (to map the GPS data with the sensory data) – where such services may be offered by different organizations. As a result, these services may lack a common communication framework, making it a difficult task for the users to obtain and integrate a comprehensive data from different services. Today, with the advent of web services, software companies are implementing services based on this new

standard. Considering the popularity and fast development of web services, a wide range of useful services can be expected in the near future. In addition, there has been a great deal of research work conducted to implement sensor network services as web services. Recognizing the potential of web service technologies, one of the members of our project group has conducted research on the feasibility of web services for sensor networks [53]. However, while implementing applications to aggregate and to collect sensory data, one must recognize that different services may belong to different domains and protecting access to these services is a fundamental system requirement.

The following is a typical scenario that will be dealt with in this thesis:

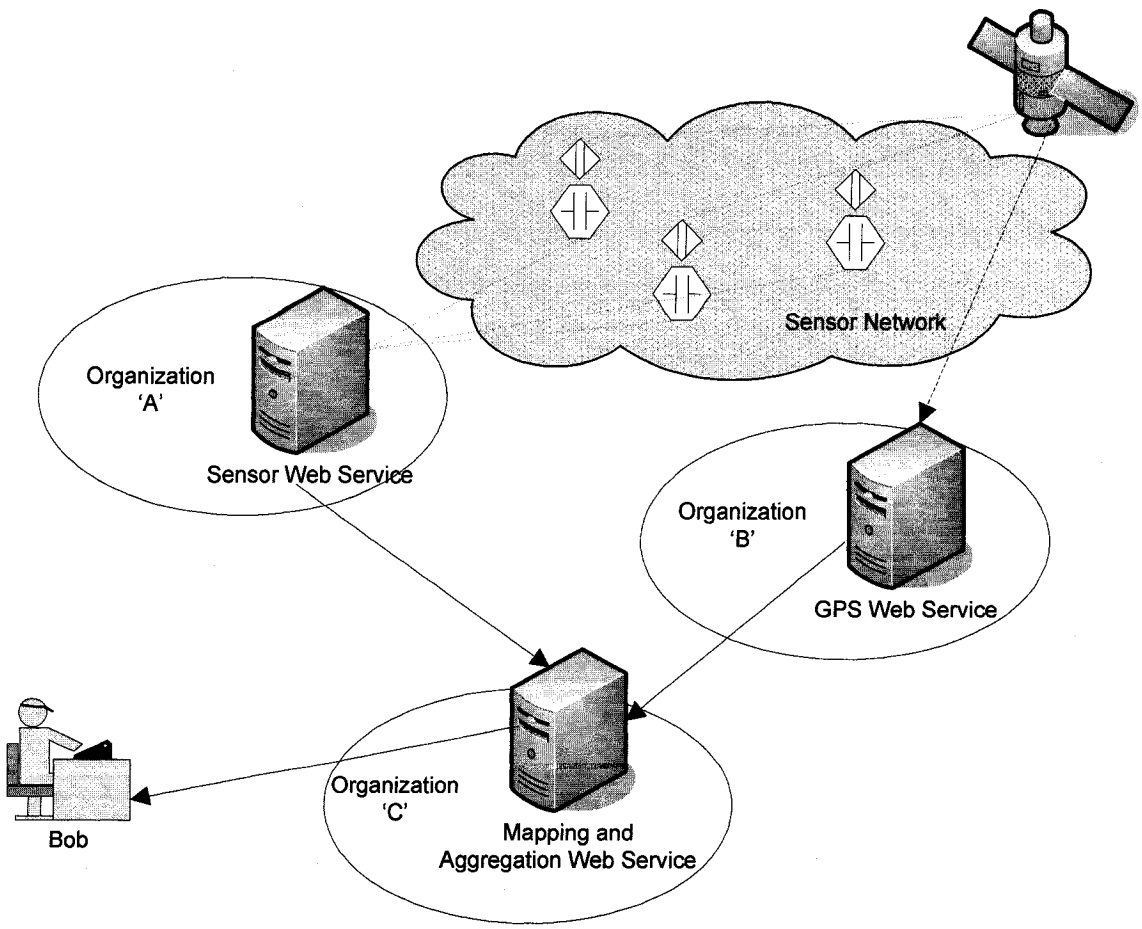


Figure 1. A Typical Sensor-Data Integration Scenario

Figure 1 is a typical example showing sensor network access through web service orchestration. In this scenario, web services are used to provide sensory data service, GPS service, and mapping/sensor searching service. Bob is a registered user for all three web services. He has different privileges at each of these domains. When Bob desires to access any of these services or group of services (for example, finding the temperature near his region) he downloads the WSDL files from web services belonging to different domains and includes them in his application. However, Bob wants to authenticate himself only once to save time and to increase efficiency. This is a simple scenario, but in reality a user may have different privileges at different organizations, and single sign-on in such a heterogeneous environment is difficult to establish and maintain. An ideal approach should be able to handle these issues with low costs and minimal administration overhead.

1.2 Problem Statement

The main problem in a federation of autonomous security domains is to maintain a balance between access rights and access restrictions. The security framework should guarantee the autonomy principle.

- *Autonomy Principle*: If an access is permitted within an individual system, it must also be permitted under secure interoperation.

On the other hand, the security framework should not violate the security of the individual system.

- *Security Principle*: If an access is not permitted within an individual system, it must not be permitted under secure interoperation.

To meet the above principles, the most popular mechanism utilized is the use of a central authority server. In this approach, the authority server authorizes the users on behalf of all the participating domains. The domains can again validate such authorization decisions and impose more local restrictions during a user's request for service access. The goal is to create a federation of resources and services for a group of users.

Creating such a federation is not an easy task. Security systems in a federation should comply with the participating domain policies, which involves high administrative cost and expertise. As the nature of the participating domains in a federation can be highly diverse, the process of access control policy integration may become highly complex. Both the users and the resources may enter and exit the federation, raising a need for maintenance of the security system. Also, the resources may belong to different autonomous domains and, hence, may implement various access control models with different levels of granularity. This thesis will treat the problem of establishing a federation and the subsequent maintenance and enforcement of access rights on users using an existing authorization framework.

1.3 Thesis Contributions

The main objectives of this research work are as follows:

- To develop a system to generate a federated access control policy in a heterogeneous environment.

- To effectively integrate the above developed system into an existing authorization framework.

The contributions of the research reported in this thesis can be summarized as follows:

- Different requirements of an authorization framework in a federated system are defined and analyzed.

- An approach for the integration of access control policies of autonomous domains in a heterogeneous environment is proposed. Such a mechanism is based on matching user-credentials associated with the roles in a RBAC model. In addition to this approach, resolution to the conflicts that may arise due to integration of the policies is discussed.

- Extensions to an existing community-based authorization framework are proposed. These include the use of the proposed policy integration approach to generate a global policy for the community and the use of standardized access control model - the RBAC model - in a community.

- An integration tool is developed based on the proposed approach of policy integration. The key aspects of the tool are the following: an administrative input request while merging the policies, detection and resolution of conflicts, and enforcement of the global policy into a community-based authority server.

The initial result from this research work has been published at the International Conference on Privacy, Security, and Trust (PST) 2006 [29].

1.4 Structure of the Thesis

In this chapter, the motivating scenario for this research work is presented. Then, the problem statement tackled in this research and the thesis contributions are presented.

The remaining chapters of the thesis will proceed as follows:

- Chapter 2: Gives the background information and description of underlying technologies. This chapter explains the fundamental standards such as RBAC, X.509, and SAML.

- Chapter 3: Defines a set of requirements for an authorization framework in a federation. Furthermore, this chapter gives an overview of previous research for the requirements defined.

- Chapter 4: Discusses the design of the RBAC policy integration tool and gives insights into the architectural and implementation decisions made for the development of the tool.

- Chapter 5: Focuses on the extension of an authorization framework. Extensions include the use of the proposed tool and of a standardized access control model.

- Chapter 6: Presents a case study in detail. Furthermore, various test results are shown for the proof of concept.

- Chapter 7: This chapter summarizes the contribution of this research work and concludes the thesis with the future research scope.

Chapter 2 Technical Background

This chapter presents the background concepts on access control, Role-Based Access Control, web services, and certificates and assertions. The objective is to provide the necessary material to set the context for subsequent chapters of this thesis.

2.1 Access Control

Access control can be defined as the protection of resources against unauthorized access. It is a process in which resource usages are regulated according to the security policy of the resource. Thus, access control is essential for controlled resource sharing among multiple users. There are two fundamental types of access control models, namely, Mandatory Access Control (MAC) and Discretionary Access Control (DAC). According to the United States Department of Defense Trusted Computer System Evaluation Criteria, MAC is a means of restricting access to the resources based on the sensitivity of the information (as represented by a label such as top-secret, moderate, etc.) contained in the resources and the formal authorization (e.g. clearance) of the subjects to access information of such sensitivity. In the DAC model, there are no explicit security-level labels, but the security domain administrator plays a significant role in assigning permissions to the users. An access to a resource is granted to a user based on the discretion of the administrator. DAC can be implemented through an access control list (ACL). Hence, the important difference between MAC and DAC is who controls the access control on an object. Recently, role-based access control (RBAC) has emerged as a promising alternative to MAC and DAC models. In a RBAC model, the concept of

roles is introduced, which are created within an organization for various job functions. The permissions to perform certain operations are assigned to specific roles. RBAC is described in greater detail in the next few sections.

2.2 Role-Based Access Control

RBAC is an access control model which is based on three sets of entities, namely, Users (U), Roles (R), and Permissions (P). A user is a person or an automated agent who can take up multiple roles. A role is a job function or title which defines an authority level. Permission is an approval of a mode of access to a particular resource. Access control policy is embodied into RBAC through relationships between RBAC entities (user-role and role-permission). Figure 2 shows an access control for a group of four resources with and without RBAC. In the RBAC model, permissions are aggregated into a role and an administrator assigns users to roles instead of assigning users directly to the permissions. In this way, the administrator has fewer relations to manage, thus reducing the administrative cost. The successfulness of RBAC comes from the fact that the notion of "role" captures the way most organizations operate. Hence, RBAC has been found to be the most attractive solution for providing security in distributed systems [18].

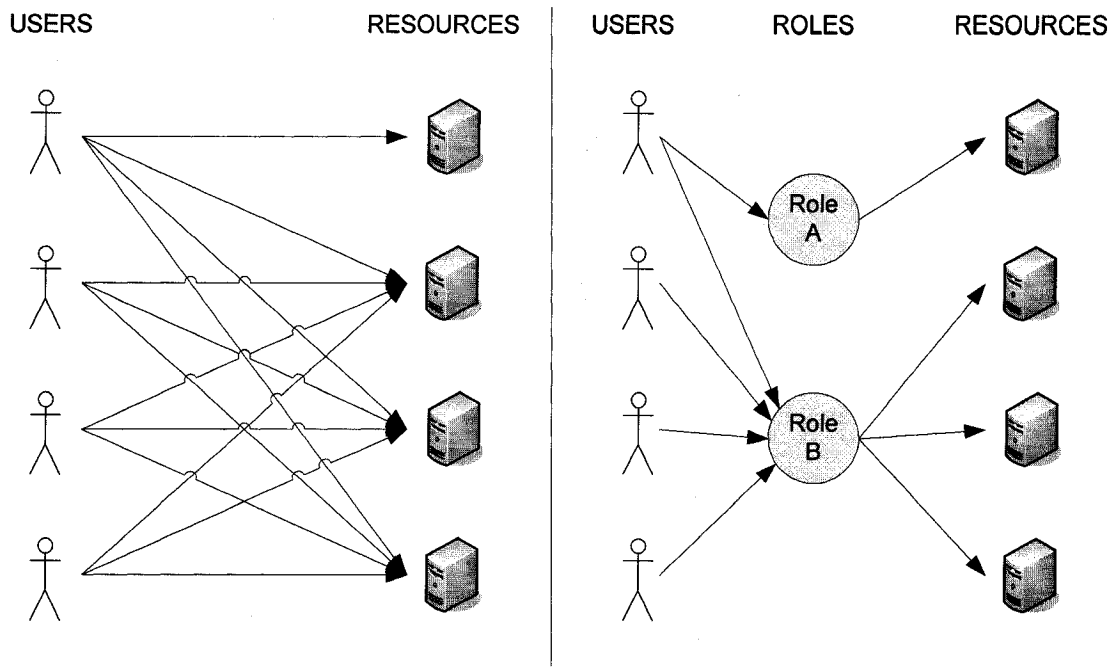


Figure 2. Access Control Implemented without Roles on the Left and with Roles on the Right

An RBAC framework can be categorized into three models:

- Flat RBAC: The basic model in which users are associated with roles and roles are associated with permissions.
- Hierarchical RBAC: The Flat model of the RBAC with hierarchical relations between the roles.
- Constrained RBAC: The Hierarchical RBAC model with constraints on user-role and role-role associations.

These three models of RBAC are described in the following sub-sections.

2.2.1 Flat RBAC

The fundamental aspect of all RBAC models is the Flat RBAC model, which consists of four basic components: *users*, *roles*, *permissions,* and *sessions*, as shown in Figure 3. A user (also termed "Subject") can be a human being, a host, or a process. A role is a function in the organization which can be played by several users. A permission is a tuple of an operation and an object. Each role may have several permissions attached to it. A session is an instance which defines the roles that are active for a user at the particular instance of connection. If a user is assigned to an administrative role, it is a good practice to activate this role only when needed. This reduces the risk of damage due to a user mistake or a session hijacking.

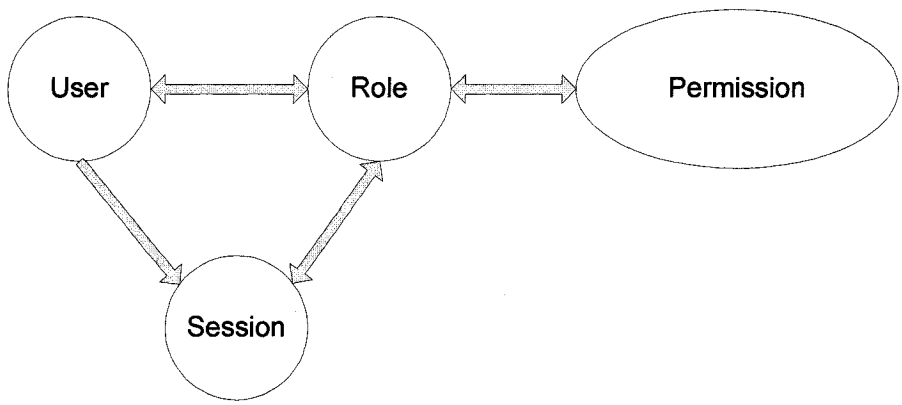


Figure 3. Flat RBAC Model

2.2.2 Hierarchical RBAC

Hierarchical RBAC extends the Flat RBAC by adding hierarchies to roles, as shown in Figure 4. In Hierarchical RBAC, if a role has child roles, it will inherit all the permissions assigned to those roles. This allows new roles to be created by combining or extending existing roles. Two types of inheritance, viz. multiple inheritance and limited inheritance,

are supported by Hierarchical RBAC. Multiple inheritance permits inheritance in multiple steps – as shown in Figure 4, the Surgeon role inherits permissions from the Nurse role. In multiple inheritance, senior roles are often considered dangerous because they aggregate too much power. Also, senior roles may inherit contradictive permissions, which can be a serious problem. However, limited inheritance may impose a limit on inheritance by allowing only one-step inheritance, thus preventing the Surgeon role from inheriting permissions from the Nurse role. The Surgeon role will inherit only the Doctor role. The two concepts can be combined together to achieve better results.

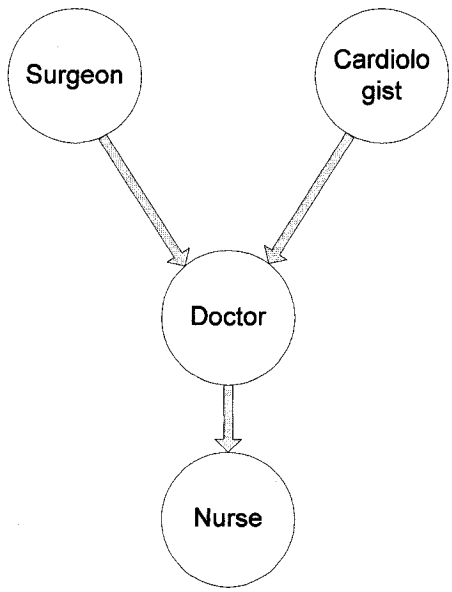


Figure 4. Inheritance Used to Combine or Extend Roles

2.2.3 Constrained RBAC

In certain situations, it is essential to impose regulations on how roles and permissions are assigned. These regulations or constraints can be static or dynamic in nature. Static constraints are enforced when a user is assigned to a role while dynamic constraints are

enforced when a user attempts to activate a role. Following are some of the common types of constraints proposed on a RBAC model [35].

- *Separation of duty* intends to reduce the risk of fraud by preventing the same user from playing two roles. For example, it is relevant in order to prevent a user from being both CEO and auditor of the same organization.

- *Cardinality* intends to limit the number of users taking up a same role. For example, an organization may not have more than one chairman.

- *Temporal constraint* is a constraint which may prevent users from playing certain roles at some hours.

2.3 Web Services

Web Services, as defined by the World Wide Web consortium (W3C), is a software system designed to support inter-operable machine-to-machine interaction over a network. It has an interface described in a machine readable format (specifically, WSDL). Other systems interact with a web service in a manner prescribed in WSDL, using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards. Web services are identified by a Uniform Resource Identifier (URI) and provide a standard means of interoperation between different applications running on different platforms. In general, the web services have the following features which make them unique:

- **Loosely coupled:** Coupling refers to the degree to which software modules depend upon each other for the software to function. Loose coupling means that the software modules do not have to be concerned about each other's internal implementation, and interact with other modules using a standard interface. In software development, coupling typically refers to the degree to which software components/modules depend upon each other. A web service can be implemented in any possible programming language, and its implementation is not visible to the user or application that calls the service. This feature enables web services to locate and communicate with each other dynamically at runtime.

- **Open standards:** Web services implement Service-Oriented Architecture (SOA), using open standards like UDDI, WSDL, and SOAP for publishing, method invocation, and data transfer. SOA is a conceptual architecture that enables dynamic E-business. The open standards are one of the most important factors contributing to the success of web services.

The foundation layer of the web services is built on three primary technologies. These are the Simple Object Access Protocol (SOAP), the Web Services Description Language (WSDL), and the Universal Description, Discovery, and Integration (UDDI).

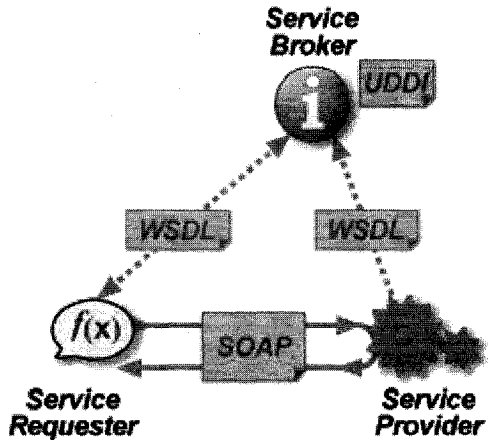


Figure 5[1]. Three Primary Technologies of Web Services: WSDL, SOAP, and UDDI

2.3.1 WSDL

Service description in web services is standardized using XML-based WSDL. A WSDL document for a web service is generally publicly accessible and provides all the information related to the web service. A WSDL document would contain supported operations and input/output message formats required to invoke the operations and the protocol bindings. A client program, after reading the WSDL document of a web service, can connect to it using SOAP protocol to actually call one of the functions listed in the WSDL. Following are the elements used by WSDL to describe a web service:

- Types: A container for data type definitions.
- Message: A definition of the data being passed in a single Remote Procedure Call (RPC).
- Operation: A description of an action (method) supported by the service.

[1] Source: *http://www.infosys.tuwien.ac.at/staff/lukasz/masterthesis_LJ.pdf, accessed on Nov., 2006*

- Port Type: A set of operations supported by one or more endpoints.
- Binding: A concrete data format specification for a particular port type.
- Port: A single endpoint defined as a combination of a binding and the network address where it can be found.
- Service: A collection of related endpoints.

2.3.2 SOAP

SOAP (Simple Object Access Protocol) is a W3C standard and a high-level communication protocol which defines the XML data flow between the server and the client. SOAP is an extensible, text-based framework enabling communication between diverse parties that have no prior knowledge of each other [57]. This is a requirement for a communication protocol in web service architecture. The three parts of SOAP include envelope, encoding rules, and Remote Procedure Call (RPC) representation. The envelope sets up a framework to indicate what the message contains. It consists of an optional header which may target the nodes that perform intermediate processing and a mandatory body intended for a final recipient. The encoding rules provide a serialization mechanism for exchanging instances of application-specific data types. Finally, RPC makes it possible to encapsulate and represent calls and responses on remote procedures.

2.3.3 UDDI

Universal Description, Discovery, and Integration (UDDI) specification provides a common set of SOAPs – API²s that enable the implementation of a service broker. UDDI is an open industry initiative, enabling businesses to publish their services and discover others services over the Internet. It is a platform-independent, XML-based registry providing access WSDL documents. A UDDI business registration consists of three components:

- White Pages: Address, contact, and known identifier.
- Yellow Pages: Industrial categorizations based on standard taxonomies.
- Green Pages: Technical information about services exposed by the business.

2.4 Certificates and Assertions

2.4.1 X.509

One of the notable building blocks of many distributed authorization systems is the X.509 standard. X.509 defines certificates that bind a public key to a person. A person in a X.509 certificate is identified by a globally unique meaningful name (Distinguished Name). Some of the main uses of a certificate include the following: digital signature, encryption, sign and encrypt, and secure file storage. Most of these applications of X.509 require the communicating entities to have a prior relationship between them. But it is impossible to guarantee such relationships in a large distributed environment. To

overcome this public key distribution problem, a Certification Authority (CA) is employed, which acts as a Trusted Third Party. The CAs are trusted and they digitally sign or issue X.509 certificates to requesting entities. VeriSign, Entrust, and Thawte Consulting are some of the public certification authorities.

2.4.2 Security Assertion Markup Language

Security Assertion Markup Language (SAML) was developed by OASIS (Organization for the Advancement Structured Information Standards) to provide a common language for exchanging security-related information between multiple domains (or companies) engaged in business-to-business and business-to-consumer transactions. Such information is called an "assertion" and belongs to an entity that may be a person or a host or any other subject. SAML defines three kinds of assertions:

- Authentication Assertions: These carry information related to the authentication acts performed by the subject to prove its identity.

- Attribute Assertions: Such assertions carry information related to the attributes of the subject, which include specific details such as citizenship or credit line.

- Authorization Assertions: These carry information related to the authorization decisions relating to the subject, which identifies what the subject can do.

As shown in Figure 6, the grey box (assertions) is standardized by the SAML specifications. However, these are the assertions produced and consumed by different authorities outside the scope of the SAML [66]. The authorities that produce the

assertions may get various kinds of data from the policy store. The model shown in Figure 6 is conceptual. In practice, different kinds of authorities may reside at a single software system. Also, not all assertions are always produced.

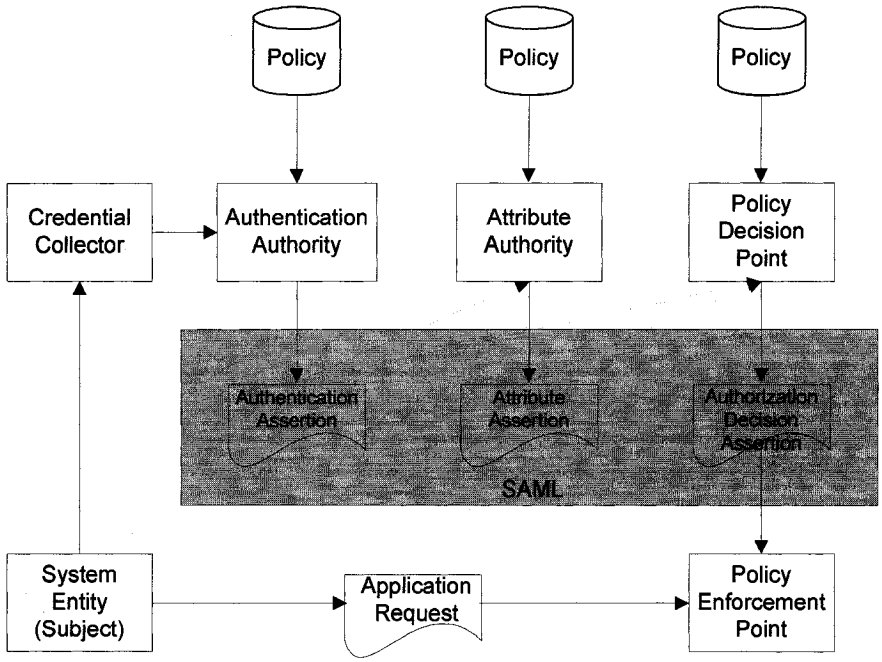


Figure 6. SAML Model for Producer and Consumer of Assertions

2.5 Definitions

Before getting into the details of access control, some of the terms required to understand this thesis are defined below.

- Coherent policy: A combined policy that is in conformity to all the individual participating policies.

- Collaboration: A process in which two or more organizations work together to achieve their independent and collective interests through a joint resource and service sharing process.

- Delegation: A mechanism by which a user delegates his task to another user.

- Doctrine: A rule in the system which is held to be true by all the entities in that system.

- Domain: A collection of entities to which applies a single security policy executed by a single authority.

- Federation: A system comprised of a number of independent domains united by a central authority.

- Fine-Grained Access Control: An access control model that provides the ability to define access policies for individuals or groups of users for actions on resources in the system.

- Policy: The set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information.

- Policy Integration/Policy Composition: A process through which a single policy is generated from two or more policies.

- Single Sign-On: A process through which users sign onto a site only once and are given access to one or more applications in a single domain or across multiple domains.

2.6 Summary

This chapter introduced the reader to access control, Role-Based Access Control (RBAC), web services, and certificates – the powerful standards currently used for business interactions – and definitions for the technical terms used in the rest of this thesis. In the next chapter, previous research in multi-domain authorization frameworks is presented.

Chapter 3 Related Works

After developing a basic familiarity with access control in the previous chapter, previous research in multi-domain authorization frameworks is presented in this chapter. The requirements for any authorization framework in a multi-domain environment are identified and described in greater detail.

3.1 Authorization in Multi-Domain

A security domain can be defined[2] as a collection of entities to which applies a single security policy executed by a single authority. In collaborations, it is quite possible to have services from different security domains aggregated to perform a useful task. However, adding security to multi-domain environments introduces challenges at many levels. In a multi-domain environment, it is difficult to specify the access rights to each user in an open distributed environment. The heterogeneity of the resources participating in the collaboration and dynamic changes in the user profiles make access control decisions complex in such an environment.

3.2 Basic Entities in Multi-Domain Authorization

In the distributed systems, authentication and authorization decisions are made by authorities who have a direct or a delegated relationship with the resource or the user or with both. The framework for such types of relationships can be implemented using trust

[2] Source: *http://csrc.nist.gov/publications/fips/fips188.html, accessed on Aug., 2006*

mechanisms, such as public key infrastructure (PKI). From this observation three basic entities in a multi-domain authorization system are defined [39].

- **Subject:** A subject is an entity named in the certificate. They can request, receive, own, transfer, and delegate certificates. A subject can be a human user, a process, or a host.

- **Resource:** In RFC 2828, a resource is defined as data contained in an information system; a service provided by a system; a system capability, such as processing power or communication bandwidth; an item of system equipment (i.e., a system component – hardware, firmware, software, or documentation); or a facility that houses system operations and equipments.

- **Authority:** An entity that is capable of issuing, validating, and revoking certificates for subjects can be defined as an authority. There are three types of authorities [36, 67]: attribute authority, policy authority, and identity authority. An attribute authority issues attribute assertions to the subjects. A policy authority issues authorization assertions based on the access control policies to the subjects. An identity authority (e.g. CA of a Public-Key Infrastructure) issues certificates that assert a mapping of cryptographic tokens to the subjects.

3.3 Functional Components in Authorization Framework

The two main functional elements in an authorization framework are the PEP (Policy Enforcement Point) and the PDP (Policy Decision Point). A PEP can be defined as a

logical entity that enforces access control policies in response to a request from a user to access the resources. A PDP makes the decision whether or not to authorize the user to access the resources based on the access control policy. A PDP can be a remote entity, but a PEP is a component residing on the resource which actually enforces the policy decisions made by the PDP. When a user requests access to a resource, the PEP at the resource may give the job of authorizing the user to the PDP. Three different types of configurations involving these two components are authorization push sequence, authorization pull sequence, and authorization hybrid sequence, which are described below.

3.3.1 The Authorization Push Sequence

In the authorization push sequence, the subject first requests an authorization authority, a PDP, for authorization assertions. The authority then issues authorization statements to the subject. These statements may be sent to the subject in the form of signed certificates or tokens. These received tokens or certificates are presented by the subject to any resource in the collaboration to which the subject wants access. The resource will accept or reject the subject's request based on the authorization statements presented by the subject. The authorization push sequence is depicted in Figure 7. Examples of such a sequence are found in many ticketing systems such as Kerberos or Keynote [6] and in the Community Authorization Service from the Globus Toolkit [58].

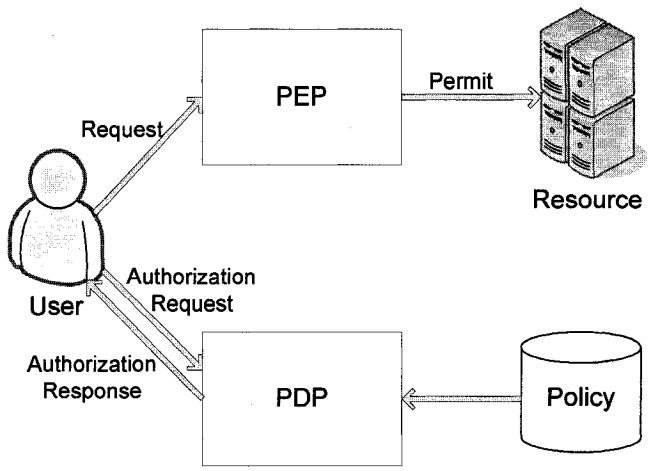


Figure 7. Authorization Push Sequence

3.3.2 The Authorization Pull Sequence

In an authorization pull sequence, the subject will directly send an access request to the resource. In order to admit or deny the subject's request, the resource sends a request to the authorization authority, the PDP, which performs an authorization decision and returns its decision to the PEP. The PEP will subsequently grant or deny the user request based on the decision from the authorization authority. The authorization pull sequence is depicted in Figure 8. Examples of this system are Shibboleth [59], PERMIS [50], and Akenti [1] authorization systems.

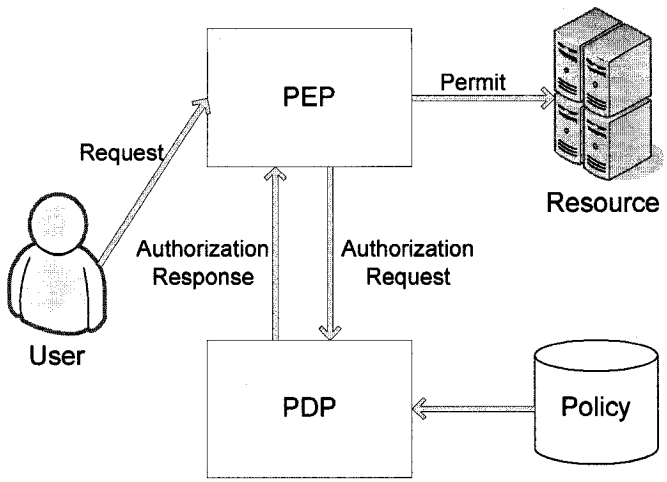


Figure 8. Authorization Pull Sequence

3.3.3 The Authorization Hybrid Sequence

The authorization sequences described in the above two sections (3.3.1 and 3.3.2) are the fundamental sequences. But there exist authorization frameworks that do not entirely match either of these two basic sequences. One such example is the combination of the pull and the push models and is termed an "authorization hybrid sequence". In a hybrid sequence, the subject initially obtains the authorization statements in the form of certificates or tokens from the authorization authority. When a subject requests access to a resource with this certificate, the PEP may query the authority in its local administrative domain to make sure that the access complies with the local security domain policies.

3.4 Authorization Framework Requirements

An authorization framework in a multi-domain environment includes a dynamic collection of resources and users unified by a common goal. Since the resources may be located in different security domains, maintaining a consistent access control policy is

very difficult because each domain may have different mechanisms for policy specification and enforcement. This situation is further complicated by the fact that the resources and subjects may be dynamic and that even the access control policies may change dynamically. Before proceeding further with authorization framework requirements, the working details of some of the popular authorization frameworks are examined.

The Community Authorization Service (CAS) [45, 46] is a software system integrated within the Globus Toolkit [58]. It allows the creation of a virtual organization (VO). It is built on the Grid Security Infrastructure (GSI) [9], which is a set of libraries and tools that focus on authentication, message protection, single sign-on algorithms, and delegation mechanisms. A VO allows administrators to manage access control in a large distributed grid network. A central CAS server is responsible for managing the policies governing access to the resources in the community. To enable this infrastructure, resource providers in the community have to delegate a subset of their policy space to the VO. The VO maintains a global policy: once users are authorized, certificates containing their credentials are issued. The users request access to the resources by presenting a certificate. The resources enforce the VO policies carried as credentials along with user requests. CAS uses the push-model of authorization sequence as shown in Figure 9.

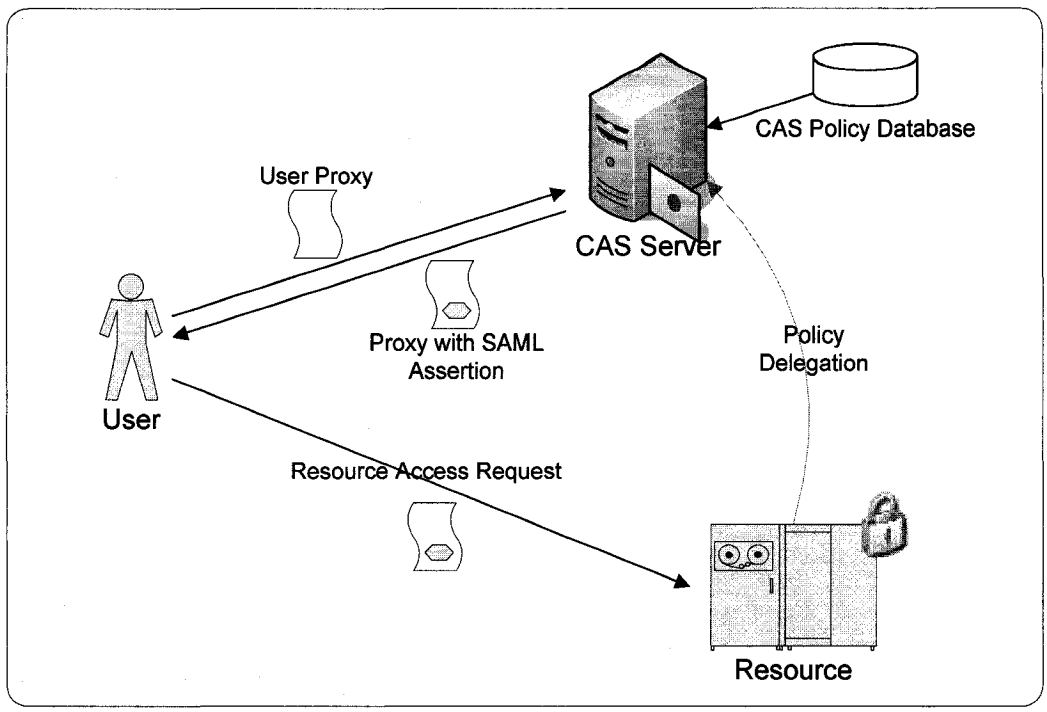


Figure 9. CAS Architecture

Note that the certificates issued to the users and containing credentials are, by default, valid for twelve hours and can be set for more or less time by the CAS server administrator. So, users can use the same certificate to access different resources in the community for the assigned period of time without having to re-contact the CAS server.

PRIMA [38], a privilege management and access control system, complements the security mechanisms present in the Globus Toolkit [17] by locating the PDP on the resource itself. PRIMA uses a hybrid authorization sequence model where the PEP queries the PDP for a high-level access decision after verifying the user privilege attributes and thereby forming a dynamic policy. A dynamic policy is the combination of the user's privileges and the resource's security policy. The PEP then interacts with security mechanisms to allocate a UNIX user account (an execution environment) with minimum access rights based on the dynamic policy. Since a user can selectively provide

his or her privileges to the resources with the access request, a least privilege access to the resources is enabled and fine-grained access control over the resources is ensured.

Shibboleth [59] is another popular single sign-on authorization framework in a cross-domain environment. It was developed by Internet2/MACE [21]. Shibboleth defines two main functional components and an optional component. The Identity Provider (IdP) creates and manages user identity, and the Service Provider (SP) controls access to resources. The third component is a "Where Are You From?" (WAYF) service to locate the IdP for a user requesting access to the resource. In Shibboleth, when a user attempts to access a resource, the SP will interact with the IdP to obtain the attributes of the user. Along with the user's attributes, the IdP should also maintain the user's attribute release policy which specifies what user attributes can be released to an external SP. Thus, Shibboleth is an example of the pull model of authorization sequence.

It is useful to define the requirements of an authorization framework in order to effectively support multi-domain environments. These requirements are as follows:

- Use of standards
- Separation of Policy Decision from Enforcement
- Access Control Policy Specification
- Access Control Policy Integration
- Integrated Policy Maintenance
- Global Policy Enforcement

In the following sections of this chapter each of these requirements is presented in greater detail.

3.4.1 Use of Standards

The adoption of security standards in an authorization framework plays an important role in allowing inter-operation between different security domains. The standards, such as SAML, XACML (eXtensible Access Control Markup Language) and RBAC, ease the authorization process between business partners. OASIS (Organization for the Advancement Structured Information Sciences) is an organization that is playing an important role in this standardization process. Many frameworks that provide access control over distributed systems, such as CAS (Community Authorization Service), Shibboleth, and PRIMA, use these security standards.

In CAS, the central CAS server grants restricted proxy certificates to community members using the X.509 standard. The newer version of CAS from the Globus Toolkit Version 3.2 [58] issues SAML assertions for authorization statements [66], embedding them within X.509 proxy certificates as non critical X.509 certificate extensions. Shibboleth is the standards-based, open-source middleware software that makes extensive use of SAML specification to provide attribute exchange. The PRIMA system uses XACML to encode privileges [38].

3.4.2 Separation of Policy Decision from Enforcement

In a multi-domain environment it is essential to separate the decision-making of an access control policy from its enforcement. This separation allows the delegation of an access control policy from the participating domain to a trusted third party.

3.4.3 Access Control Policy Specification

According to Pearlman et al. [48], one of the key problems associated with the formation and operation of distributed virtual communities is the specification of access control policies. Many of the existing security systems have been developed with a specific access control policy in mind.

In recent years, RBAC has emerged as a powerful access control model. Due to its inherent richness in modeling organizational behavior such as hierarchy, separation of duty, and cardinality, it provides a promising approach to support uniform representation of policies across business domains. Moreover, RBAC is also capable of modeling traditional discretionary access control and mandatory access control models [46]. So, in the rest of this section, different ways to represent RBAC are discussed.

A very different, but intuitive, approach to security policy representation is the graphical specification described by Koch et al. in [34]. In this approach, roles, users, sessions, and permissions are denoted as nodes. Edges between the nodes of a graph represent the association between the entities in RBAC model. The following figure (Figure 10) shows an example of a graph-based formalism of RBAC.

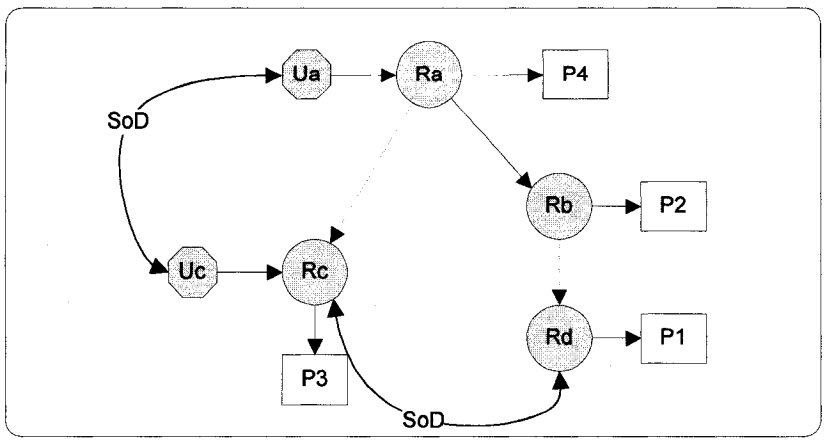


Figure 10. The RBAC Graph Model

In the above figure, the graph defines all the possible edges that may exist between two nodes. An edge from a user node Ua to a role node Ra represents the membership of Ua to Ra. An edge between the two nodes Ra and Rb represents the role hierarchy. The inheritance hierarchy and activation hierarchy in the RBAC model is represented by solid and dashed lines, respectively. Separation of Duty (SoD) can also be specified in the graph model.

Another promising approach to uniform representation, sharing, and dissemination of information across domains, is XML technology. X-RBAC is an XML-based RBAC policy specification language [3, 23]. In addition to specification of RBAC policies, X-RBAC provides a framework for specifying mediation policies between multiple RBAC domains [25]. In the X-RBAC model, the user, role, and permission specifications are kept separate from their assignments to allow the independent design and administration of policies, and hence support a modular RBAC model. XML schemas are defined to capture each component of the RBAC. Thus, information about users, roles, and permissions are available from the corresponding XML sheets: XML User Sheet (XUS), XML Role Sheet (XRS), and XML Permission Sheet (XPS). The assignments between

these elements are captured using an XML User-to-Role Assignment Sheet (XURAS) and an XML Permission-to-Role Assignment Sheet (XPRAS). An XPRAS specifies the permission set attached to a particular role, whereas an XURAS specifies the user credentials required for the user to be assigned to a particular role. If no explicit user-credentials are specified for a role, then any user can take up the role, which could be a guest role. The following figure represents the X-RBAC policy components.

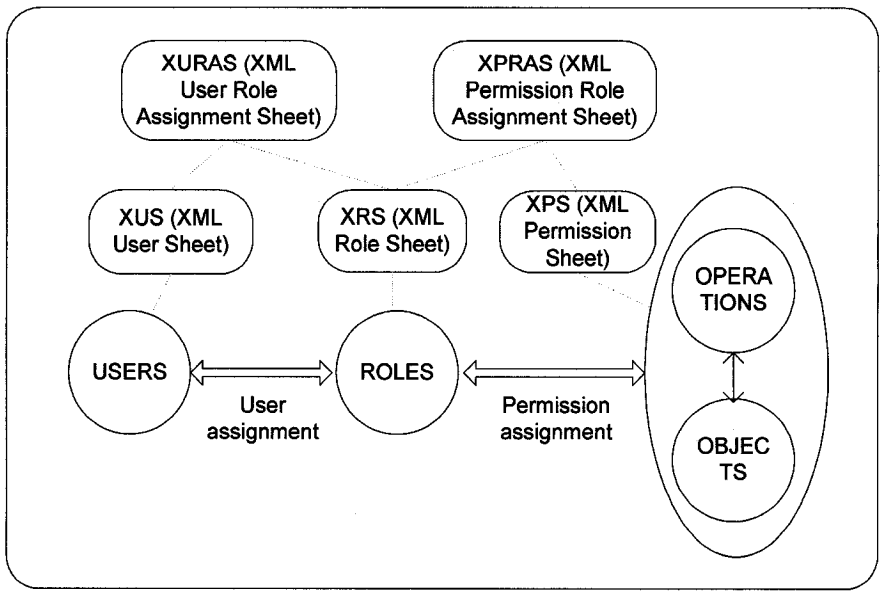


Figure 11. X-RBAC Policy Components to Define Extended RBAC Elements

3.4.4 Access Control Policy Integration

Policy integration is a mechanism by which a global security policy is formed that remains consistent with the participant domains. Single Sign-On (SSO) in a multi-domain environment may require the participating security domains to integrate their security policies. A single global security policy is essential for consistent access control across domains. Currently, much of the integration is left to the security administrators. This

may not be an optimal solution, as integration in these cases involves many factors, such as policy complexity, privacy considerations, and administrator expertise. An inconsistency between integrated global security policies and participant domain policies could lead to improper access of resources and security holes in the system.

In [8], the problem of combining independent access control policies specified in different languages is addressed. The algebra of security policies together with its formal semantics is proposed. To make the algebra compatible with a large number of policy specification languages, the algebra is designed to operate only on three sets - the subjects (S), objects (O), and actions (A) of the policies. Hence, authorizations terms in a policy are the triples in S x O x A. The semantics of algebraic expressions is a function that maps each expression of the policy onto a set of the above-mentioned basic authorization terms. The policy composition operators (e.g., Addition (+), Conjunction (&), Subtraction (-), Closure (*)) are applied over the authorization terms in order to integrate the policies. Such an approach will have heterogeneous policy support, policy language expressiveness, and different abstraction level support [8]. In [5], Bidan et al. proposed a solution for specifying the security policies in terms of access control rules and information flow rules. A set of operators are defined for combining these independent policies into a global security policy. The above proposed textual composition framework has some drawbacks. The design of the algorithm becomes complicated and difficult to manage and certify [24]. Another drawback of this mechanism is that this composition framework requires the participant organizations to reveal their confidential policies [40].

Nicole Dunlop et al. [13] proposed a scalable policy model that is able to cope with the high rate of changes in large, evolving enterprises. This model is suitable for a large, heterogeneous, and evolving enterprise as it supports dynamic roles, dynamic policies, and dynamic conflict resolution. This approach uses new concepts including enterprise domain, policy space and policy authority. Subsequent work from Dunlop et al. [14] addresses the problem of conflict detection in a dynamic environment. The model discussed in [12, 13] is suitable for evolving enterprises and is not adequate for existing independent policy composition.

PEACE is a security framework for Policy-based Establishment of Ad-hoc Communities [31]. In this approach, the doctrines are introduced into the community, which define the roles in the community and the characteristics needed by the participants in order to be eligible to play the role. The proposed security framework is well suited for small and well-defined ad hoc communities, but it is a difficult task to define doctrine for a large set of resources and users from multiple security domains.

To ease the process of integration, access control implementations at the participating domains needs to be standardized. Thus, for policy integration, a formal access control system that can capture the set of controls in each domain is needed. In this respect, as mentioned in the previous section, RBAC is based on three sets of entities: users, roles, and permissions. Roles in an RBAC model represent various job functions in the system, and hence, most of the security policies can be represented in RBAC model. Thus, RBAC supports a uniform representation of security policies. However, relatively less work has been done on RBAC policy integration, and most of the work assumes the basic notion of

matching permission sets for mapping the roles. The rest of this section summarizes related work in the field of RBAC policy integration models.

Shafiq et al. [56] defined a set of relations between inter-domain roles for the composition of a consistent RBAC policy. In this proposed framework, the RBAC policies of individual domains are specified using an RBAC graph-based specification model [34]. The authors define Policy Integration Requirements (PIR) to be satisfied by RBAC policy integration techniques. The PIR defined in [56] are

- Element preservation: The elements (role, user, and permission) in the composed RBAC should have a corresponding element in the input RBAC policies.

- Relationship preservation: A relationship between input RBAC policy elements should be preserved during integration.

- User authorization preservation: A user's authorization in his or her home domain should be preserved after integration.

- Order independence: The order in which RBAC policies are integrated should not influence the resulting RBAC policy. If the final outcome of the policy integration mechanism depends on the order in which policies are combined, then one must find an integration order that gives maximum interoperation with minimum overhead. However, restricting the order may not be an attractive option since in most collaborative environments, domains join or leave a collaboration at any time.

- Constraint satisfaction: The constraints of the input RBAC policies must be satisfied by the resulting integrated RBAC policy.

In [56] and [25], inter-domain roles are mapped by comparing the permission set associated with the corresponding roles. These permission sets are matched using a schema integration technique [69] from the database area [56]. The major drawbacks of the proposed role mapping technique are as follows:

- Difficulty in supporting the heterogeneity of the participating domains: The permission set in RBAC is a set of tuples of objects (resources) and operations that can be performed on the objects (e.g., read, write, and modify operations on a file). If the resources in the participating domains are of dissimilar types, then the operations on these resources may differ substantially. Permission set-based composition models depend on the homogeneity of the permission sets attached to the mapping roles, and therefore the integration technique cannot be feasible in a heterogeneous environment.

- Difficulty in respecting privacy policy of the participating domains: The policy integration of multiple domains is highly influenced by the privacy policies of the constituent organizations. But the proposed integration models require that the collaborating organizations share their permission sets before integrating the policies. This could have major implications on the privacy policy of the organizations because it is essential that an organization adheres to its privacy policy during collaboration.

Another approach for secure interoperation between multiple organizations with RBAC policies is SERAT: Secure Role Mapping Technique [57]. This paper proposes a novel approach of access paths for secure interoperation. However, initial interoperability is enabled by creating cross-links that inter-connect domains. Cross-links are either explicitly permitted or restricted links, decided by the collaborating domain administrators. Cross-link formation could be a complex task and depends on administrator expertise and their privacy policy.

It appears that very little research has been done with respect to policy integration between different security domains with RBAC policies. In general, most of the existing approaches for RBAC policy composition are based either on a permission set-based role mapping or an explicit administrator-specified mapping, which has major drawbacks.

3.4.5 Integrated Policy Maintenance

The policy lifecycle model includes various activities such as identifying, specifying, analyzing, enforcing, maintaining, and, finally, deleting policies. Policy maintenance is becoming more important for managing distributed environments such as virtual enterprises. Changing the environment requires an adjustment in the relevant access control policies to the new situation. The environmental changes include deploying new services, introducing new users, removing the existing users or services, etc. Hence, after integrating the local security domain policies, it is important to maintain the integrated global security policy. The integrated policy should be flexible enough to support changes in the user profiles or resources within the collaboration. Such changes in the collaboration should not affect other participating resources or users in the collaboration.

For example, in CAS [51], a community administrator can explicitly change a user profile without affecting the participating domains.

3.4.6 Global Policy Enforcement

The mechanism to enforce the integrated global policy into the participating resources should be transparent to the users and the resources to the greatest extent possible. Also, users should be able to access services in the collaboration without the knowledge of the domain to which they belong. CAS authorization framework [51] maintains a VO, where services are modified to enforce the CAS credentials. A user should present CAS credentials from accessing any service in the VO. As a first step, a CAS-enabled service enforces the resource policies for the VO by using the identity of VO and not the identity of an individual user. The VO policies towards the user, as expressed in the CAS credentials, are enforced on the user. Optionally, resources can further enforce any additional policies regarding the user. Thus the effective access control policy enforced on the user is shown in the below figure [51];

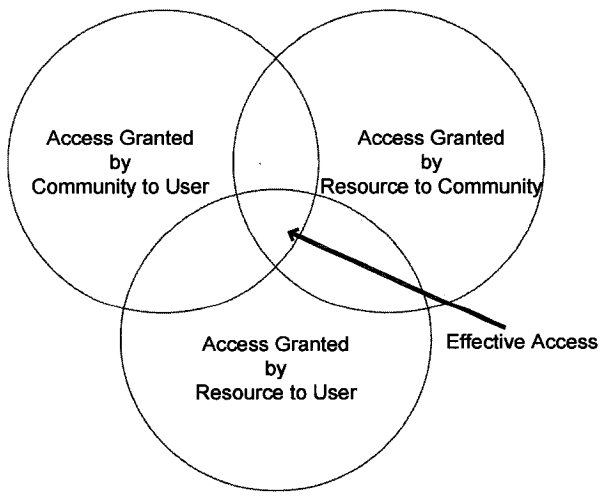


Figure 12. Effective Access in CAS

As shown in Figure 12, the access rights granted to a user are the intersection of access rights granted by the resource to the community, the access rights granted by the community to the user, and, finally, the optional access rights granted by the resource to the user.

3.5 Summary

In summary, this chapter reviewed some of the popular authorization frameworks. The requirements for an efficient authorization framework in a multi-domain environment were identified. These requirements were then discussed in greater detail. It appears that little research work has been conducted on security policy integration techniques. Moreover, the popular CAS framework does not include any of the policy integration mechanism; instead, it depends on CAS administrator expertise to integrate the security policies. Table 1 summarizes the requirements defined in this chapter with the related research on each of them.

Requirements of an authorization framework	Related research (Brief)	Focus of the thesis
Use of standards	• SAML • XACML • RBAC Use of these standards in different authorization frameworks	Use of existing standard
Separation of policy decision from enforcement	Separation of policy decision from its enforcement to allow the delegation of the access control policy	Use of PEP and PDP in the authorization framework
Access control policy specification	• Graphical specification of RBAC • XML specification of RBAC	Policy specification suitable for multi-domain environment and policy composition model
Access control policy integration	• Algebraic model for policy composition • Scalable policy model for large evolving enterprises • Policy establishment in Ad hoc communities • RBAC role-permission based policy integration model • RBAC inter-domain role mapping using access paths Description of each of the above existing models of policy composition with their merits and demerits	Focus on proposing a new policy integration model to overcome the drawbacks identified from the existing models
Integrated policy maintenance	The integrated policy should be flexible enough to support changes in the user profile or resource within the collaboration	Support for dynamic environment
Global policy enforcement	Description of Virtual Organizations by Community Authorization Framework	Global policy enforcement to web service technologies

Table 1. Summary of Defined Requirements for an Authorization Framework and Related Work

The following chapters of this thesis propose a policy integration technique, and a proof-of-concept implementation is also described.

Chapter 4 RBAC Policy Integration Tool

In this chapter, the requirements of an efficient policy integration tool are defined and analyzed. From the analysis, the generic architecture and design details are presented with an illustrative example. Finally, the chapter is concluded with implementation details.

4.1 Assumptions and Requirements

Due to the immense popularity of the RBAC model, the proposed policy integration tool assumes that the RBAC model is used at each of the participating security domains. At present, identity authority services (e.g. issuing certificates, validating certificates, revoking the certificates, etc.) are provided to the public by organizations such as VeriSign, Cacert, and Thawte Consulting. However, from the motivating scenario described in section 1.1 of Chapter 1 of this thesis, it is envisioned that authorization services for the collaborating environments can be provided in a similar fashion as identity authority services. The authorizing authority can be a Trusted Third Party (TTP) that can bootstrap the collaboration between different businesses. Hence, an authorization service may act as one of the many services available for the users in the collaboration. During the collaboration bootstrapping, security domains have to delegate their RBAC policies to the TTP, which integrates these RBAC policies to form a single global policy. From the literature review in Chapter 3, the requirements that should be confirmed by the tool to integrate RBAC policies are defined below.

- Assignment of integrated RBAC roles to the users: In the RBAC model, users are assigned to particular roles by a security domain administrator. But in a collaboration, a TTP administrator should assign integrated RBAC roles to users, who may wish to join the collaboration. Assigning roles to users is usually based on specific conditions that need to be satisfied by the users. Such user-to-role assignment conditions at the TTP should be consistent with those of the participating domains.

- Participant Privacy Policy: A participating security domain may not reveal the privileges associated with a role in its domain. For example, consider a university as one of the participating domains that has a Professor role in it. The Professor role may have a permission to access all the conference rooms in the university. During collaboration, the university may wish to share its Professor role, but without publishing what permission a Professor role has within the university.

- Environment Heterogeneity: One key challenge in the composition of independent security domain access control policies is its diverse resources with dissimilar operations applied to them. This heterogeneity may also arise because of policy representation mismatch and semantic heterogeneity of policies among participating security domains.

Thus, it is a TTP's responsibility to generate and maintain a global security policy that is consistent with all the participating security domain policies. From the literature review in Chapter 3, it was observed that TTP functionalities are very similar to CAS server functionalities in a community authorization framework. However, a TTP may not

impose its own policies (community policies) on the users participating in the collaboration as is possible in CAS. Also, a TTP administrator may not be a collaboration administrator and can therefore be unaware of the type of resources and users participating in the collaborations. Therefore, access control policy maintenance can be a cumbersome task for a TTP. Maintenance of an access control policy includes adding and removing users and resources from the collaborations.

4.2 Design of Policy Integration Tool

4.2.1 Analyzing the requirements

The requirements for a RBAC policy integration technique identified in the previous section are analyzed in this section to facilitate the design of a policy integration tool:

- Assignment of integrated RBAC roles to the users: The TTP administrator will be able to assign collaboration policy roles to the users only when the participating domains define and delegate the conditions for a user to be assigned to a role in their RBAC policy. A review of specification of RBAC policies in Chapter 3, suggests that the X-RBAC model is more suitable than other types of RBAC specifications for extracting such conditions from the security domains. X-RBAC extends the standard NIST RBAC model, including XML-based policy components, to specify the constraints in assigning users and permissions to the roles and user activation of roles in sessions. XURAS is a policy component in X-RBAC, which specifies the attributes a user should have in order to take up a role.

If the participating domains delegate the XURAS policy component, then the TTP administrator can assign new users to roles based on the conditions specified in XURAS.

- Participant Privacy Policy: In the context of the RBAC model, the participating domain may not disclose the permission set associated with the roles during collaboration. Permission sets are represented by XPS and XPRAS policy components in an X-RBAC model. XPS is an XML-instance document for permission specification, and XPRAS is an XML-instance document associating a role with the permissions. Hence, policies should be composed without XPS and XPRAS components.

- Environment Heterogeneity: Collaboration between different domains occurs when useful services aggregate together for a group of users. In the collaboration, the diversity is vast among the participating resources and the services, rather than among the users accessing these services. Since all the services are conglomerated for a set of users, the heterogeneity among users in the collaboration is least. For example, a graduate student (user) can act as Intern (role) in an enterprise that has limited privileges (permission set) associated with the Intern role. However, the same graduate student may take up a Student role having more privileges in the university domain. When both the university and the enterprise domains need to collaborate for a partnership research project, it is easy to capture the equivalency between the Student role and the Intern role based upon their associated user credentials, i.e., the attributes the user is required to have to take up the role. This observation indicates that the policy integration mechanism can minimize the

heterogeneity in collaboration by considering the users accessing the services rather than the services or resources themselves.

From the above analysis it can be concluded that X-RBAC can be effectively used for RBAC policy integration in a multi-domain collaborative environment. Moreover, the policy integration technique should not be based on permission set matching (XPS and XPRAS policy components of X-RBAC) in order to avoid privacy policy breach. But it should be based on users (represented by XUS and XURAS policy components in X-RBAC) to maximize the efficiency of integration by minimizing the heterogeneity.

4.2.2 Generic Architecture

Figure 13 shows the generic architecture of the policy integration model, wherein the participating domains export their X-RBAC schema-based policy components, namely, XRS and XURAS. However, it is assumed that the meaning of user-credentials is same across all participating domains. The TTP forms the global security policy by integrating the imported domain policies. The users are issued authorization statements from the TTP and they can use them in resource access requests within the collaboration. The goal is to create a general authorization framework that supports federation.

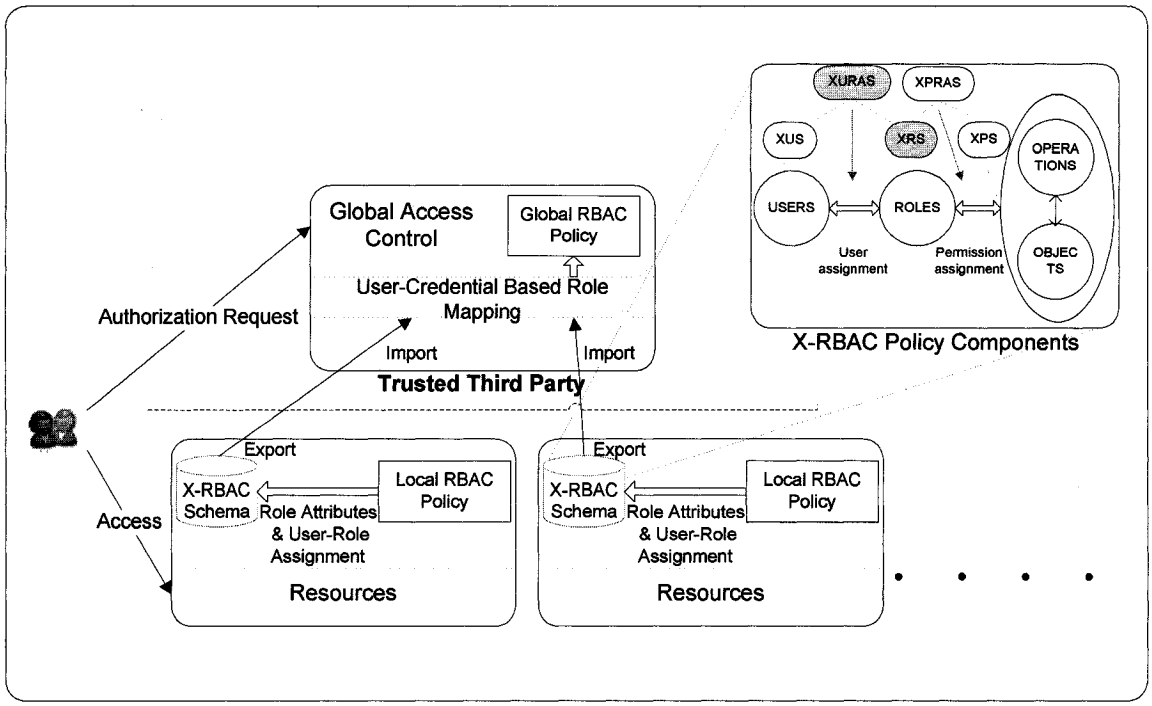


Figure 13. Generic Architecture for the User-Credential Based Policy Integration Model

The X-RBAC policy thus generated at the TTP is partial. The global policy does not contain XPS and XPRAS policy components, so the privileges associated with roles are unknown at the TTP. Only the participating domains understand the roles and the privileges associated with the roles.

4.2.3 RBAC Policy Integration

In the RBAC context, access control policy integration involves defining a mapping between cross-domain roles. By virtue of this role mapping, any user authorized for a role Ra in domain A may get access to all permissions of a mapped role Rb in domain B, depending on the type of role mapping.

4.2.3.1 User-Credential Based Role Mapping

By applying the analysis results from the previous sections, a user-credential based role mapping for RBAC policy integration is proposed. A user credential is a set of attribute-value (AV) pairs that need to be satisfied by the user in order for the user to be assigned to the corresponding role. For instance, the set of AV pairs associated with a Volunteer role at an IEEE conference may include (student status = enrolled full-time student) AND ((member status = IEEE student member) OR (registration status = conference-registered)). Logical expressions such as AND or OR can be used between AV pairs to define a complete credential set. In the example, a security administrator at the conference can assign the role of Volunteer to a person who is a full-time student and is an IEEE student member or registered at the conference. In the X-RBAC model, XURAS, a policy component, specifies the user attribute or credential for the roles in RBAC.

In a user-credential based role mapping, the AV pairs associated with the roles are matched against each other to define a mapping between the cross-domain roles. From the above example, if the conference domain is collaborating with one of the sponsorers, "University A," to share the information, then the security policies of both the domains need to be merged. Let us consider that University A has a role named "IEEE Member" for students who are members of the IEEE student branch at the university and who have special permissions for a few auditoriums at the university. AV pairs associated with an IEEE Member role may include ((student status = enrolled full-time student) AND (institute = University A)) AND (member status = IEEE student member). The comparison of user credentials for the roles of Volunteer and IEEE Member should yield

a mapping between these two roles. The mapping should be such that any user who can take the role of the IEEE Member should be able to take up the role of the Volunteer at the conference, but a volunteer at the conference may not have sufficient privileges to take up the IEEE Member role in the university domain.

Based on the user credential comparison, different types of role mappings can be defined. If Ra is a role in a domain A and Rb is a role in a domain B, then comparing user-credential sets associated with roles Ra and Rb results in any one of the following five types of role mappings.

1. Role Ra contained in Role Rb: Role Ra is said to be contained in role Rb if the user-credential set associated with the role Ra, UC_{set}(Ra), is a subset of the user-credential set associated with the role Rb, UC_{set}(Rb). A role mapping is defined between role Ra and role Rb such that Rb is junior to Ra. Figure 14 depicts the user-credential set matching that results in a role mapping.

Formally, $\forall$ uc | uc $\in$ UC_{set}(Ra) $\subset$ UC_{set}(Rb).

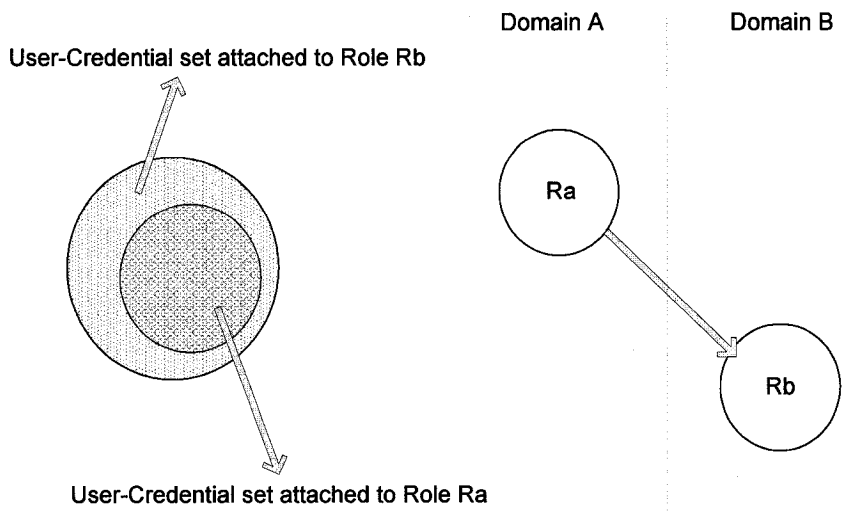


Figure 14. Role 'Ra' Contained in Role 'Rb'

2. Role Ra contains Role Rb: Role Ra is said to contain role Rb if the user-credential set associated with the role Rb, UC_{set}(Rb), is a subset of the user-credential set associated with the role Ra, UC_{set}(Ra). In this case, Ra is a junior role to Rb in the integrated policy. The user-credential set comparison and the resulting role mapping is shown in the following Figure 15.

   Formally, $\forall$ uc | uc $\in$ UC_{set}(Rb) $\subset$ UC_{set}(Ra).

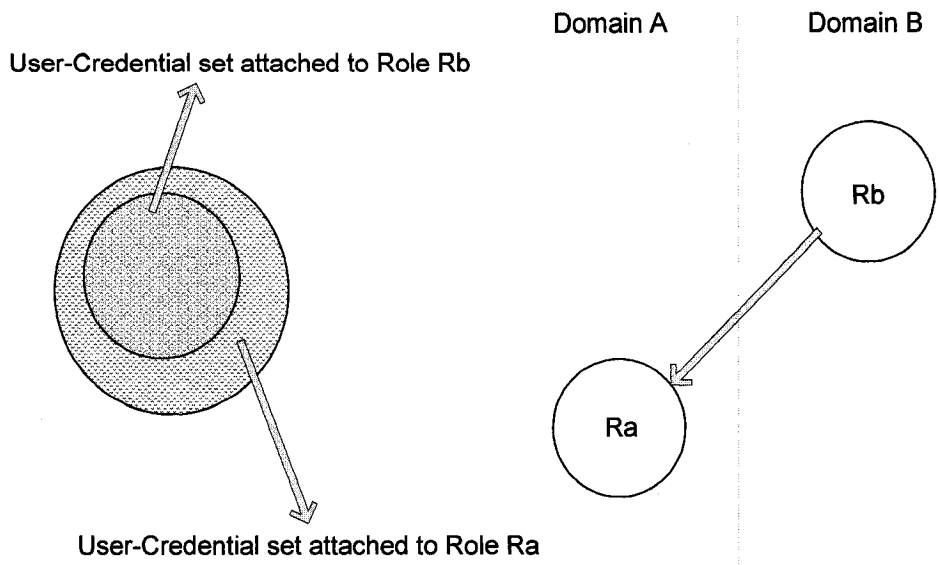


Figure 15. Role 'Ra' Contains Role 'Rb'

3. Role Ra is equivalent to Role Rb: Roles Ra and Rb are said to be equivalent if Ra is contained in Rb and Rb is contained in Ra. Both of these roles are merged into one single role, Rc, during the integration process, as shown in the following Figure 16.

   Formally, $UC_{set}(Ra) = UC_{set}(Rb)$.

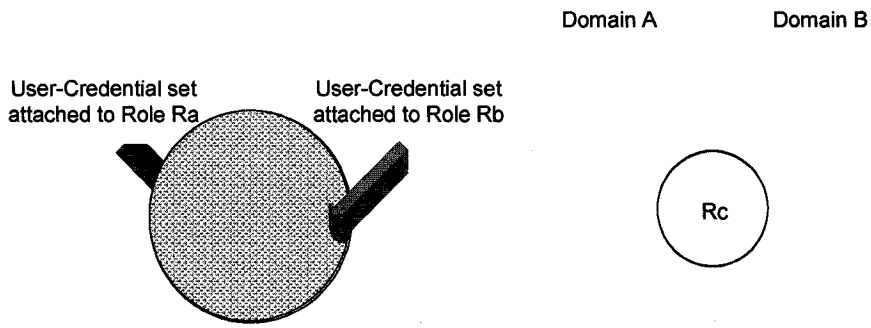


Figure 16. Role 'Ra' is Equivalent to Role 'Rb'

4. Role Ra intersects with Role Rb: Roles Ra and Rb are said to intersect with each other if some of the user credentials associated with Ra, $UC_{set}(Ra)$, match with some of the user credentials associated with role Rb, $UC_{set}(Rb)$. In the integrated policy, a new role, Rc, is created which is junior to both roles Ra and Rb and is associated with subset of matching user credentials of both the senior roles.

   Formally, $\exists$ uc | uc $\in UC_{set}(Rb) \Rightarrow$ uc $\in UC_{set}(Ra)$.

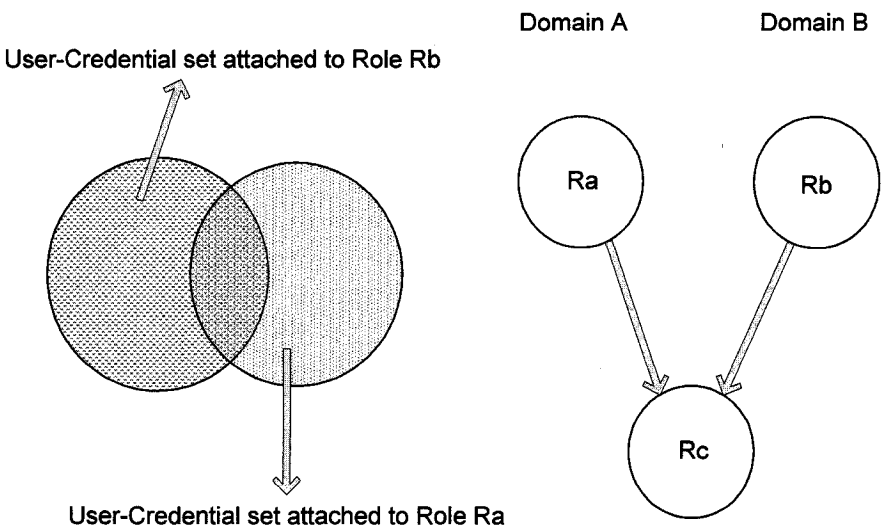


Figure 17. Role 'Ra' Intersects with Role 'Rb'

5. Role Ra is non-related to Role Rb: Roles Ra and Rb are said to be non-related to each other if the user-credential set associated with role Ra does not match with the user-credential set associated with role Rb. No mapping is defined between roles Ra and Rb in the integrated policy.

   Formally, $UC_{set}(Ra) \cap UC_{set}(Rb) = \emptyset$.

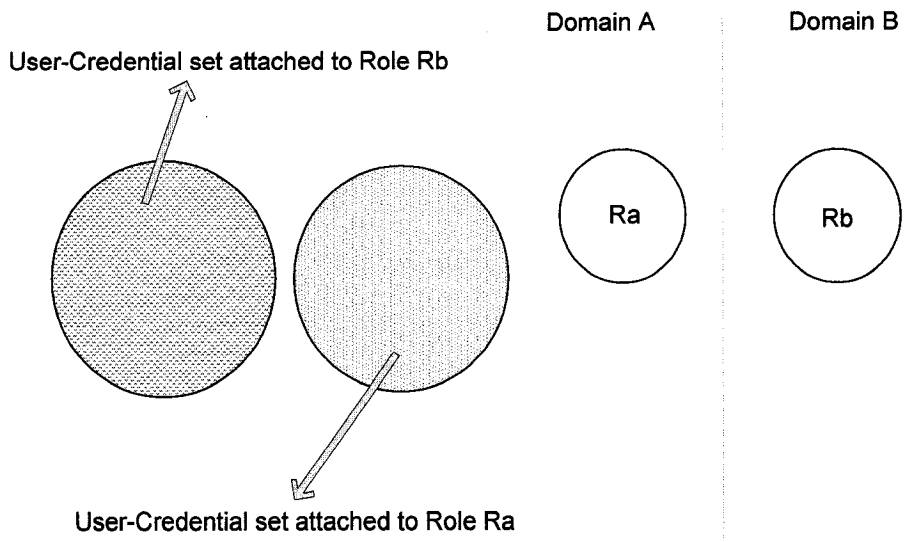


Figure 18. Role 'Ra' is Non-Related to Role 'Rb'

4.2.3.2 Heterogeneity in Composition

One key challenge in the composition of the user-credential set associated with roles is resolving heterogeneity among them. Various types of heterogeneity need to be addressed during a policy integration process, namely:

- Attribute naming conflicts: These types of conflicts arise because of the use of different names to represent the same user attributes. They generally arise in heterogeneous environments as participating domains construct their security policies independently of each other. Naming conflicts can be tackled using the synonym-based comparison approach. Moreover, it is assumed that the meaning of user credentials is the same in all domains. If the conflicts cannot be resolved, then the tool can request input from the security administrator.

- User-credential expression conflicts: Conflicts among expressions arise when the hierarchical level and sequence of AV pairs forming the user-credential set associated with the roles differ from one another. Such conflicts can be resolved using regular expressions, which can describe the complex expressions in a compact way. In the process, each different AV pair is assigned a unique literal, and the literals thus assigned to all the AV pairs are grouped together to form a regular expression representing a user-credential set associated with a role. For instance, from the previous example of role-matching, a Volunteer role having a user-credential set (student status = enrolled full-time student) AND ((member status = IEEE student member) OR (registration status = conference-registered)) is represented as "(a(b|c))", where literal "a" represents the *AV* pair (student status = enrolled full-time student), "b" represents (member status = IEEE student member) and "c" represents (registration status = conference-registered). The alternation operator, the vertical bar "|", represents the logical function "OR". Similarly, an IEEE Member role with the user-credential set ((student status = enrolled full-time student) AND (institute = University A)) AND (member status = IEEE student member) is represented as "(adb)', where literal "d" represents the AV pair (institute = University A). Once represented as regular expressions, the user-credential sets associated with the roles to be mapped can be easily searched for common patterns. In the case of volunteer role and IEEE member role matching, the regular expression "a(b|c)" is a subset of "(adb)", since regular expression "a(b|c)" represents expressions 'ab' OR 'ac' and expression 'ab' is a part of the regular expression "(adb)". This makes an IEEE member a parental

role to the Volunteer role in the integrated RBAC. But it is quite possible to have overlapping operators. In such cases, security administrator input is requested. For example, the AV pair (level >= undergraduate) and (level = graduate) are overlapping. In such cases where the attributes are the same but the operators ("=" and ">=" in the example) and the values ("undergraduate" and "graduate" in the example) are different, security administrator input is requested to decide the mapping between the two AV pairs.

Heterogeneity in the user-credential set can also be solved using a schema-integration technique from the database area. However, one of the assumptions made in the user-credential based role mapping technique is that all the local domains export their user-credentials for roles in the form of XURAS and XRS – XML sheets. For these sheets, XML schemas are defined by the XRBAC model. Hence, all the local domains follow the same XRBAC schema for user-credential specification. The schema integration problems, as defined in [69], are classified into two categories: naming conflicts and missing attributes. Further, the problems are compounded by little knowledge of local schemas, a large number of local schemas, and fast-evolving local schemas. Since it is assumed that all the participating domains use the XRBAC model for user-credential specification, the problem is reduced to naming conflicts and missing attributes (or user-credential expression conflicts).

4.2.3.3 Conflict Resolution

The policy integration algorithm can combine the RBAC policies from several independent security domains. The merging of RBAC policies might create conflicts in

the global policy. Such conflicts may arise due to role hierarchy between the roles. Moreover, while matching the user credential set in the proposed algorithm, it occasionally requests administrator input, which may also be a factor for the inconsistencies in the global security policy. One type of inconsistency that may arise is cyclic inheritance in which hierarchical relations in the global policy may form a cycle that enables a user in a lower level of the role hierarchy to assume permissions of a higher level. Figure 19 shows a cyclic inheritance due to inter-domain role mapping in which a user associated with role Ra3 in domain A can access the permissions of his superior role Ral due to the merging of two domain policies, namely, domain A and domain B.

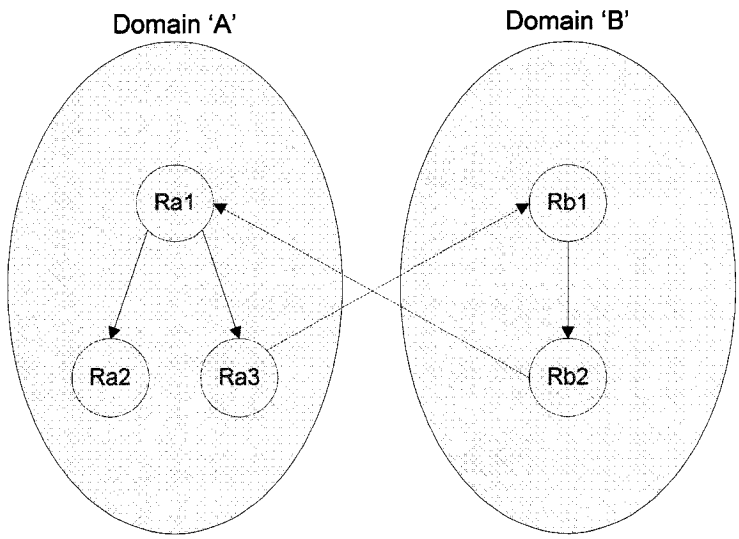


Figure 19. Policy Inconsistency Due to Cyclic Inheritance

The solution to this problem is to remove one of the inter-domain role mappings generated during the policy integration phase. In the above example, one of the inter-domain role mappings, 'Rb2'->'Ral' or 'Ra3'->'Rb1,' should be removed. However, such removal may significantly reduce the inter-operability between the domains. Hence, the tool requests the security administrator's input showing the links that need to be

deleted to overcome the conflict. Thus, the inter-domain conflicting links between roles are deleted based on an administrator's input.

Another type of conflict that may arise in a multi-domain environment is the accessibility increase for the users in their home domain due to the collaboration. Accessibility increase in a participating domain is due to newly-created inter-domain role mapping during the integration process. Following are two cases of accessibility increase:

- *Deduced mapping between local roles*: The role mapping between two intra-domain roles that is deduced due to inter-domain role mappings. Figure 20(a) is an example of deduced role mapping wherein equivalency mapping between the roles Ra and Rx and the roles Rb and Ry deduced an intra-domain role mapping between the roles Ra and Rb.

- *Violation of limited inheritance*: Figure 20(b) is an example of violation of limited inheritance. Limited inheritance may prevent the Ra role from inheriting permissions from the Rc role. However, Ra will inherit all the permissions of the Rb role. In the integrated policy, direct inheritance is introduced between Ra and Rc due to inter-domain role mappings between the roles Ra and Rx and the roles Rc and Ry.

Such deduced role mappings can be removed by deleting the inter-domain role mapping. Hence, it is a tradeoff between the level of inter-domain access and the level of increase in accessibility, depending on the administrator's decisions.

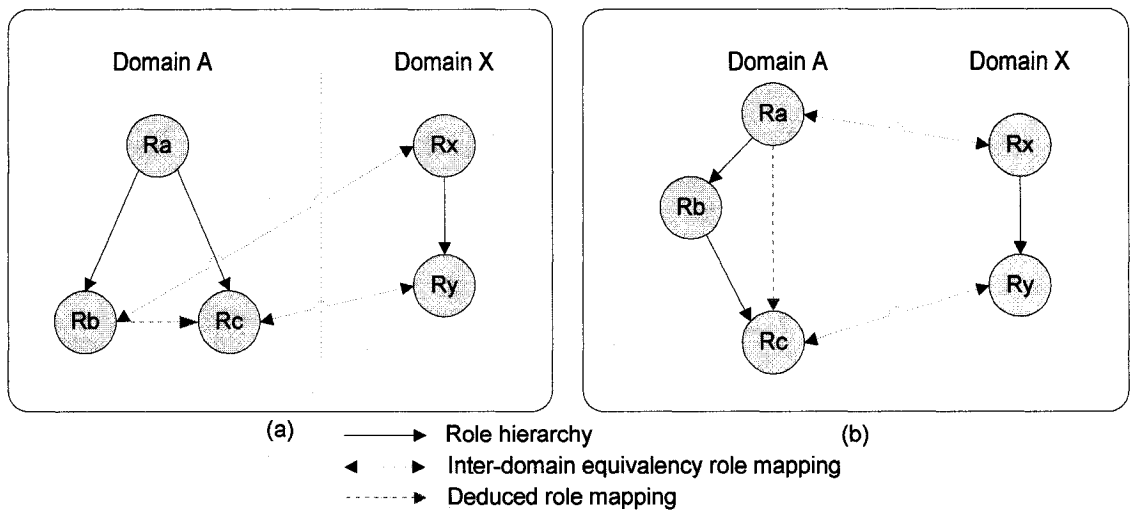


Figure 20. Deduced Intra-Domain Role Mappings

4.2.3.4 Illustration

In this section, a detailed example of the merging of two autonomous domains, namely the Astronomy Lab (AL) and the Scientific Computational Society (SCS) is discussed. The RBAC policies for the above-stated domains are intuitively described using graphs in the following Figure 21.

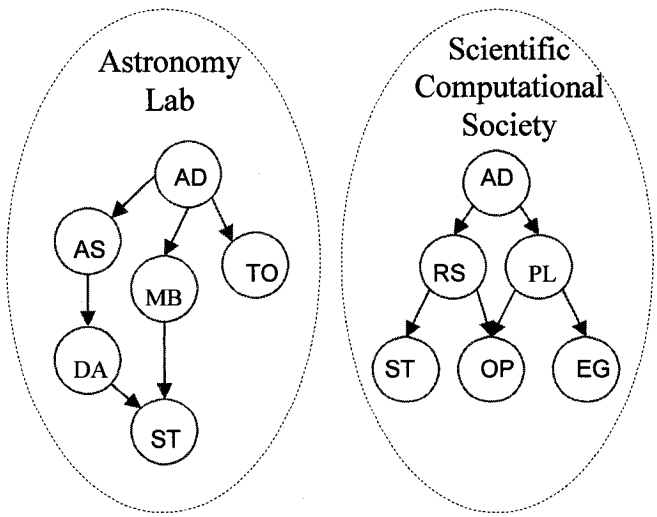


Figure 21. RBAC Policy Graphs for Two Autonomous Domains AL and SCS

Figure 21 depicts a graph-based formalism of the RBAC policies for the domains AL and SCS. The above-mentioned RBAC policies need to be specified by the security administrator from each domain. In this graph-based model, roles are represented as nodes, and edges represent the inheritance relationship between the roles. Inheritance relationships allow the users of a senior role to inherit all the permissions of the junior roles. The capital letters within the nodes in Figure 21 represent the role labels whose names and associated user credentials are given in Table 2. The services provided by the AL to its users include different levels of access to its astronomical data while the SCS provides access to its computational resources. When a group of scientists needs to work with both of these services, they should invite these services for collaboration. The user-credential sets associated with the roles are given in Table 2.

DOMAIN	ROLE	LABEL	ATTRIBUTE VALUE PAIRS
AL	Administrator	AD	User = Dr. X Qualification = Professor AND Organization = University 'A'
AL	Astronomer	AS	Position = Faculty AND (Department = Astrophysics OR Certification = AL)
AL	Member	MB	Member = Recognized Astronomy Labs
AL	Telescope Operator	TO	Qualification = Professional Engineer
AL	Data Analyst	DA	Qualification = Engineering AND Degree = Bachelors
AL	Student	ST	Student = Recognized University AND level > Undergraduate)
SCS	Administrator	AD	User = Dr. X Qualification = Professor AND Organization = University 'A'
SCS	Project Leader	PL	Member = SCS
SCS	Researcher	RS	Designation = Professor
SCS	Student	ST	Student = Recognized University AND level > Undergraduate
SCS	Operator	OP	Qualification = Professional Engineer AND Branch = Computer Engineering
SCS	Engineer	EG	Qualification = Engineering

Table 2. Attribute Value (AV) Pairs Associated with Roles

For collaboration, XRS and XURAS policy components from both of the domains are delegated to the TTP. The TTP generates a single policy that is coherent with the AL and SCS domains. Figure 22 depicts the generated global policy as an RBAC graph. It should be noted from Table 2 that the user credential for the Researcher role is (Designation = Professor) and that of the Astronomer role is ((Position = Faculty AND Dept =

Astrophysics) OR (Certification = AL)). From the user-credential matching technique it can be concluded that user-credentials (Designation = Professor) and (Position = Faculty) are one and same, since a professor is a position in the faculty. However, a Researcher role does not contain all of the credentials associated with an Astronomer role. So, the user-credential set of a Researcher role is a subset of the user-credential set of an Astronomer role, which makes an Astronomer role a parent role to a Researcher role in the integrated policy. Since the parent can take on the role of the child due to their qualifications, a user who can take up the role of Astronomer in the AL domain can now take up the role of Researcher in the SCS domain. Also, the Data Analyst role in the AL domain is merged with the Engineer role in the SCS domain as they both have the same user-credential set.

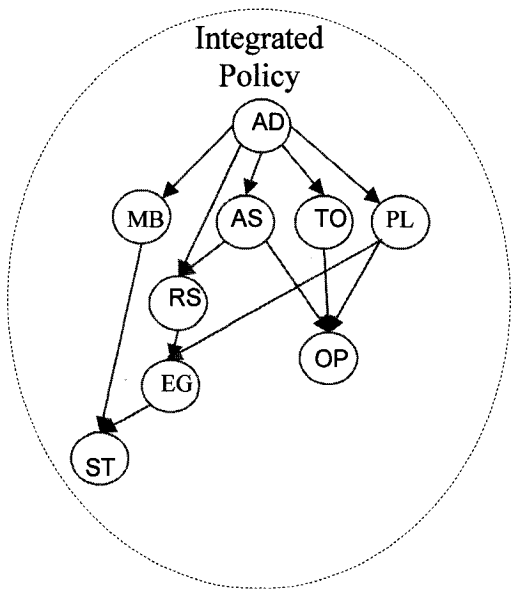


Figure 22. Integrated RBAC Policy

4.2.3.5 Policy Integration Algorithm

Tables 3 and 4 depict the formal steps of the proposed algorithm. The algorithm, *Policy-integrate*, takes XRS and XURAS policy components of participating domains as input and generates XRS and XURAS policy components of the integrated policy. It should be noted here that other components of X-RBAC, such as XPS and XPRAS, are not supplied as input or generated as output by the tool, as they are not needed for the functioning of the TTP in assigning and authorizing the users in the collaboration. The algorithm iteratively integrates two RBAC policies into one. In the first iteration, the algorithm calls a procedure *Policy-map* to combine Policies 1 and 2. In the subsequent iterations a new participant domain policy is combined with the policy integrated in the previous iteration.

The procedure, *Policy-map*, integrates two policies by generating role mappings based on the user-credential set associated with the roles and verifies the integrated policy. *Policy-map* calls *Role-map* to map the roles. *Role-map* uses the top-down approach in comparing the roles of two domains to establish one of the five types of role mappings. For defining a mapping between two roles, each of the AV pair of the user-credential set is compared. Such comparison is done by generating a synonym set for the attribute, value, and operator in the AV pair. Semantic coherence between such a synonym set is determined and evaluated. If a decision cannot be made, then an administrator input is requested. Once the AV pairs are mapped, regular expressions are generated to express the whole set of user credentials. These regular expressions are evaluated against each other to map the inter-domain roles.

Algorithm: Policy-integrate (P_1, P_2,...P_n)
1: P = {XRS[P_1], XURAS[P_1]}
2: **for** i ← 2 to n
3: P_1← P
4: P_2← P_i
5: P ← Policy-map (P_1, P_2)
6: **return**

Algorithm: Role-map (r1, r2)
1: **for** each uc1 ∈ UC_{set}(r1)
2: **do if** uc2 ∈ UC_{set}(r2) **and** not-compared-previously (uc2)
3: re1 ← Compare-user-credential (uc1, uc2)
4: **for** each uc2 ∈ UC_{set}(r2)
5: **do if** uc1 ∈ UC_{set}(r1) **and** not-compared-previously (uc1)
6: re2 ← Compare-user-credential (uc2, uc1)
7: regular-expression-match (re1, re2)
8: **if** re1 ∩ re2 = Ø
9: **then return**
10: **else if** contain (re1, re2)
11: **then** link (r2, r1)
12: **else if** contain (re2, re1)
13: **then** link (r1, r2)
14: **else if** overlap (re1, re2)
15: **then** create-role (r)
16: link (r, r1)
17: link (r, r2)
18: UC_{set}(r) ← UC_{set}(r1) ∩ UC_{set}(r2)
19: **else if** contain (re1, re2) **and** contain (re2, re1)
20: **then** create-role (r)
21: UC_{set}(r) ← UC_{set}(r1)
22: delete-role (r1)
23: delete-role (r2)
24: **return**

Table 3. Policy Integration Algorithm

Algorithm: Policy-map (P_1, P_2)
1: **for** each r1 $\in P_1$
2: **do if** r2 $\in P_2$
and not-compared-previously (r2)
3: $P_1 \leftarrow$ Role-map (r1, r2)
4: **for** each r2 $\in P_2$
and not-compared-previously (r2)
5: $P_1 \leftarrow$ include (r2)
6: **if** cyclic-inheritance (P_1)
and deduced-role-mapping (P_1)
7: **then** input $\leftarrow$ administrator-input ()
8: delete-link-in (P_1, input)
9: **return**

Algorithm: Compare-user-credential (uc1, uc2)
1: literal $\leftarrow$ 'a'
2: synonym-map ({attr1, op1, val1} $\in$ uc1, {attr2, op2, val2} $\in$ uc2)
3: **if** contain (uc1, uc2) **and** contain (uc2, uc1)
4: **then** $uc1_{literal} \leftarrow$ literal
5: $uc2_{literal} \leftarrow$ literal
6: **else if** contain (uc1, uc2)
7: **then** $uc2_{literal} \leftarrow$ literal
8: $uc1_{literal} \leftarrow$ literal + "|" + increment (literal)
9: **else if** contain (uc2, uc1)
10: **then** $uc1_{literal} \leftarrow$ literal
11: $uc2_{literal} \leftarrow$ literal + "|" + increment (literal)
12: **else if** overlap (uc1, uc2)
13: **then** $uc1_{literal} \leftarrow$ literal + "|" + increment (literal)
14: $uc2_{literal} \leftarrow$ literal + "|" + increment (literal)
15: **else if** uc1 $\cap$ uc2 = $\emptyset$
16: **then** $uc1_{literal} \leftarrow$ literal
17: $uc2_{literal} \leftarrow$ increment (literal)
18: **else**
19: **then** input $\leftarrow$ administrator-input ()
20: **return**

Table 4. User-Credential Mapping Algorithm

4.3 Implementation

This section deals with the implementation of the policy integration tool, the development tools, and libraries involved in implementation. Basically, the tool takes X-RBAC policy components as its input and applies the generated policy to an authorization authority, which is discussed in detail in the next chapter. The system-level diagram showing the main components of the tool is depicted in Figure 23.

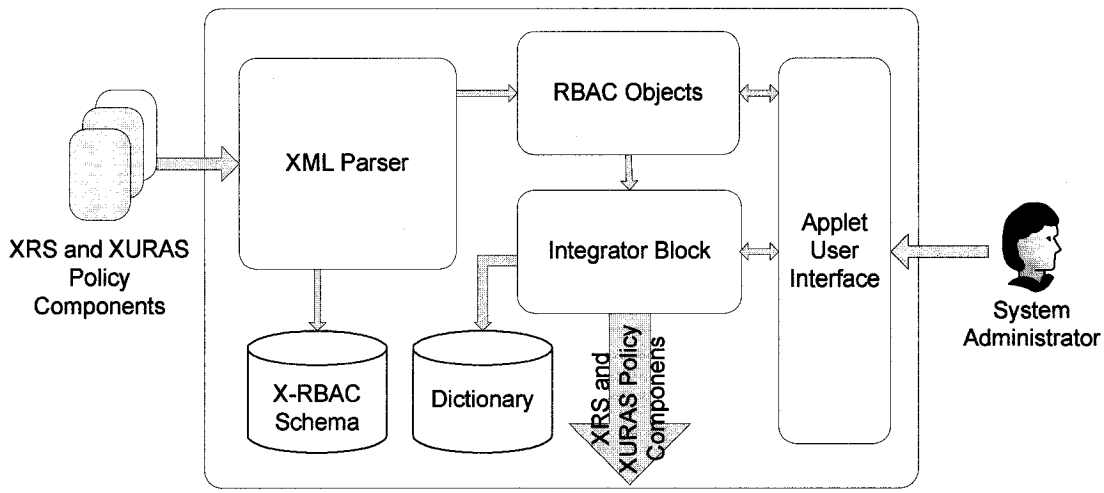


Figure 23. High Level System Diagram of the Integration Tool

As shown in Figure 23, the XML Parser parses the input policy files with the help of X-RBAC schema files. Once parsed, RBAC objects are created for each domain. The integrator block can access the generated RBAC objects and dictionary database, thus producing the integrated policy components. The applet user interface module implements various applet windows for the user interface. The front-end of the tool is a Java applet-based implementation for the user interface. A Java applet is used as front-end to facilitate the integration of policies on the security server from a remote place. The

back-end of the tool is built on many libraries for policy integration and is programmed in Java language.

4.3.1 Front End

Figure 24 shows the main Java applet window representing the front-end of the integration tool. At present, the integration tool includes mechanisms to integrate policies, check and eliminate inconsistencies in the integrated policy, and enforce the policy into CAS (discussed in detail in Chapter 5). The tool can simultaneously integrate up to five independent policies in a single run to keep the applet small and neat. More policies can be integrated by running the tool iteratively. The security administrator can select the input level (high, medium, and low) based on which administrator input is requested.

Figure 24. Main Applet Window of Integration Tool

4.3.2 Back End

Eclipse 3.1.2 was the development environment of choice for the integration tool back-end. It is a very powerful integrated development environment (IDE). Figure 25 shows the UML diagram for the Java classes implementing the core integration functions. The base of all the classes is the *RoleMap* class. This class implements the basic functionalities required for the policy integration. The *ParserRS* class implements the XML-parsing functionalities–along with the classes *XMLXURAS* and *XMLXRS* for XURAS policy components and XRS policy components respectively. The classes *RBAC, Role, Credential,* and *URA* implement the functionalities of the RBAC model.

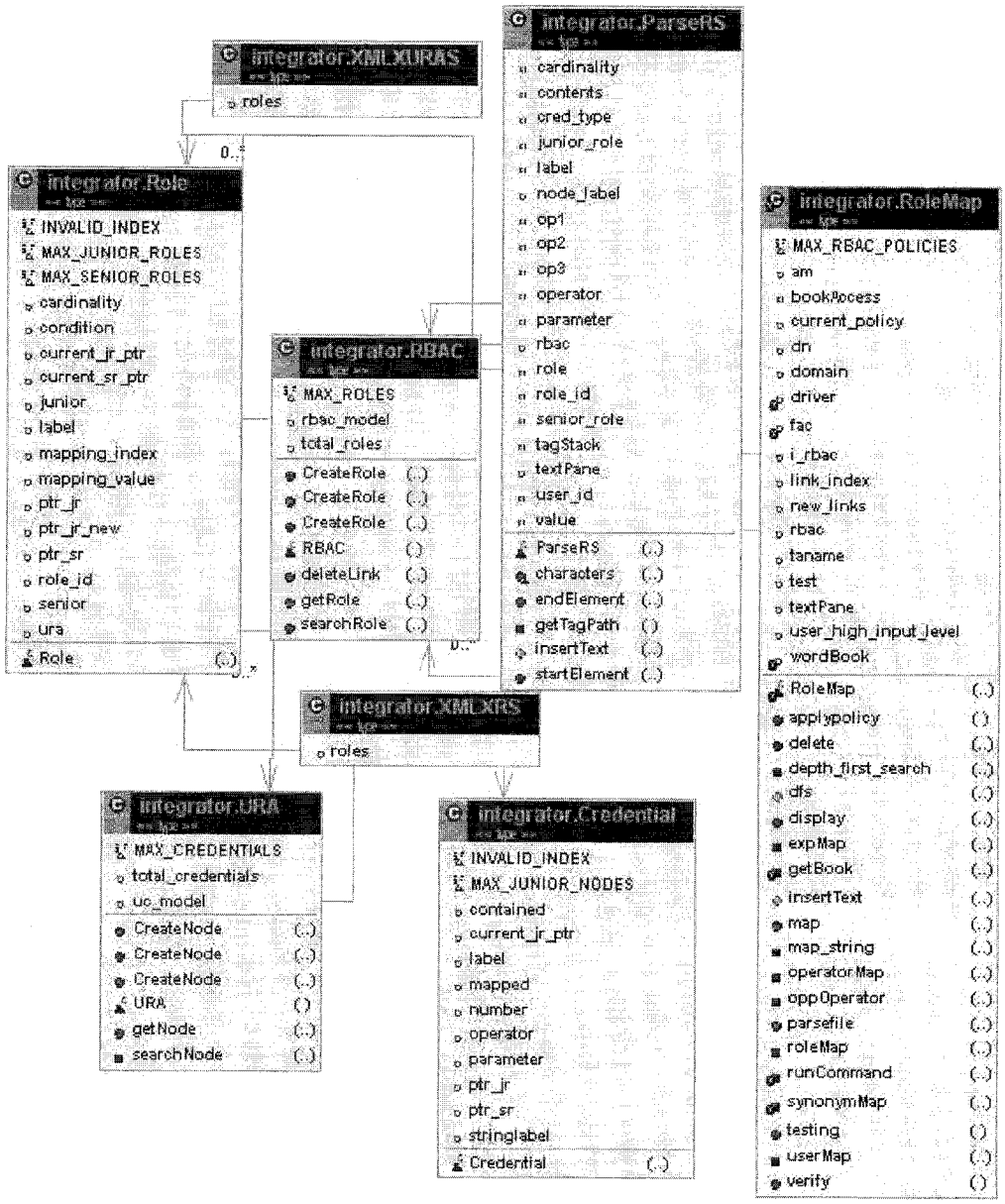


Figure 25. UML-Based Class Diagram for the Policy Integration Classes

4.4 Summary

In this chapter, a new approach for the integration of RBAC policies from heterogeneous domains was proposed. The proposed mechanism represents a fundamental shift in the

traditional paradigm of comparing and merging RBAC policies based on role privileges. XML-based RBAC specification language (X-RBAC) is used to capture the authorization policies of participating domains. The proposed approach allows the roles from different domains to be mapped based upon user-credential sets associated with the corresponding roles. By using this approach, domains will be able to inter-operate seamlessly in heterogeneous environments. The approach ensures that collaborating domains do not violate their privacy policies. This is because no assumption is made about the role privileges in any collaborating domains.

Chapter 5 Extensions to Community Authorization Framework

The community authorization framework maintains a virtual organization for the resources and the users, distributed over multiple security domains. In this chapter, extensions to the community authorization framework are discussed. These include the integration of the proposed policy composition tool into the CAS server and the addition of a security component to web services that can act as potential resources in the collaborations. The integration of a policy composition tool into a CAS server would mean applying the integrated X-RBAC policy components into a CAS database. This will enable the CAS server with a tool-generated global access control policy. Adding a security component to a web service enables it to be a CAS-enabled resource. A CAS-enabled resource should enforce the community access control policy for the users requesting access to the resource. These extensions to the CAS help to build an authorization framework for the collaboration of diverse web services from different domains.

5.1 CAS with Policy Integration Tool

The current implementation of a CAS does not provide any mechanism for integrating the policies of the resources participating in a community. A community administrator needs to insert each security policy statement using the commands provided by the Globus Toolkit for configuring the CAS server. However, the policy statements entered

should be coherent with the participating domains, which depend largely on the community administrators and their expertise. Such a configuration is an administrative overhead and a time-consuming process. Explicit configuration of the access control policy in a CAS server may lead to security holes in the collaboration further leading to improper access of resources depending on the complexity of the policies. Hence, the integration of the proposed policy composition tool into CAS would enhance its functionality by eliminating inconsistencies in the policies of the participating domains.

5.1.1 Overview of Access Control Policy in CAS

The CAS service is deployed as a part of the standard Globus toolkit (starting from version 3.2) for installation on any UNIX-flavored operating systems. Currently, this service is not available on a Windows™ platform. The CAS service is not loaded at start-up but can be enabled by updating the *server-config.wsdd* file with the *loadOnStartup* property. Once deployed, the CAS service needs to be configured with the description of the VO and the back end database to store the access control policy for the community. The database should support a JDBC driver and SQL queries (for example, PostgreSQL). The Globus Toolkit is provided with the bootstrap properties necessary to create tables in the database, but once bootstrapped; the database needs to be initialized with policy statements by the security administrator (or the "super user," as defined by the Globus group). The detailed description of the CAS configuration procedure is presented below [58]. Analyzing the procedure helps the integration of the X-RBAC model into the CAS server using the commands provided by the Globus toolkit.

- Creating a user group: The security administrator should create user groups before adding any user into the community. Access rights are granted to the user groups; hence, these rights indirectly get granted to all the users in the user group. Thus, user groups in the CAS model invariably represent the roles in the RBAC model. To create a new user group the cas-group-admin command is used.

- Adding a trust anchor: A community represents the collaboration of resources and users from different domains. Hence, users may be authenticated by different identity authorities (or "trust anchors", as described by Globus). It is necessary to add the identity authority in the database before adding the users in order to enable authentication of the users. It is usual that each domain may have an identity authority to authenticate the users in its domain. However, the user and the security administrator can also be identified by the same identity authority (as in the case study in Chapter 6).

- Creating an object group: An object or resource group is created to manage the resources. It is quite possible in a community that many resources of the same type (FTP servers) implement the same access control policy. Such a set of resources can be efficiently managed by grouping them together and granting access rights on the entire group rather than on individual resources.

- Adding a resource to an object group: Once an object group is created, the security administrator can add resources into the object groups. Each resource or object is associated with a namespace in the CAS server. A namespace defines certain features of the resource, such as the base URL that should be prefixed to

objects that belong to a namespace. For example, a namespace for an FTP server resource can be *FTPDirectoryTree* which is added to the CAS server at startup.

- Creating and adding an action mapping: An action object represents explicit access rights such as a read operation on a file object. All the relevant action objects (for example, read, write, and modify) can be stored together in a service type. User groups are explicitly granted action objects for the predefined resource groups.

In summary, the configuration of CAS server includes creation of user groups, object groups, and service types, which can be directly mapped to roles, objects, and operations of the RBAC model. As shown in Figure 26, a service type and object group together represent the permissions; i.e., the mode of operation permitted on specific resources. The user groups are associated with the permissions, thus making them comparable to roles in a RBAC model. However, CAS policy does not explicitly support a hierarchy of user groups or constraints on user groups and, hence, can only implement the flat RBAC model. There are two different methods to incorporate hierarchical RBAC model into the CAS. In the first method user group objects are granted membership of the role – say, Ra (or action object) – and membership of all the junior roles (or action objects) to the role Ra. In an alternative method, each participating domain needs to maintain hierarchical information of the roles and thus can authorize the user based on his or her role membership – say Ra – in the community, the hierarchical level of the role Ra in the community, and the mapping of the role Ra to the local role. Thus the similarity of the CAS policy model with the RBAC model can be exploited to integrate the policy composition tool.

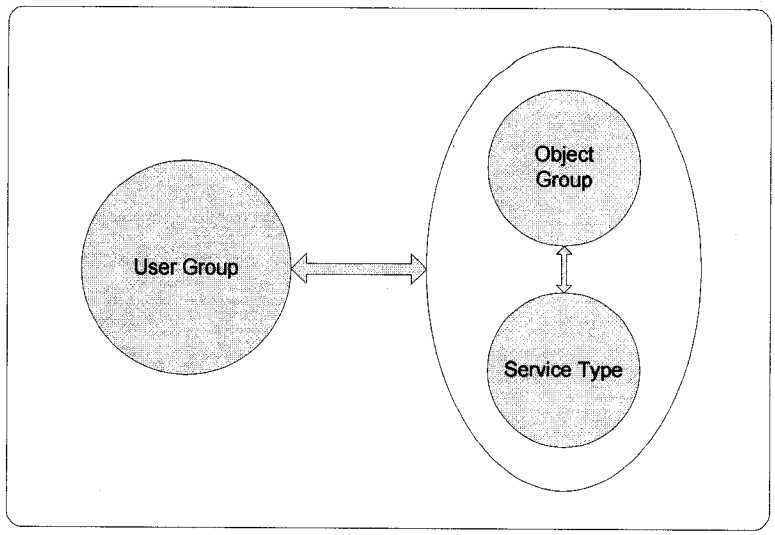


Figure 26. CAS Access Control Model – Similarity with the Flat RBAC Model

5.1.2 CAS with RBAC Model

Although the CAS policy model was not explicitly developed for RBAC, it was designed to be flexible. Besides, analysis in the previous section has shown that the CAS policy model is very similar to the flat RBAC model. To handle the RBAC model in a community framework, without fundamentally changing the existing mechanisms and related tools, the following modifications should be incorporated into the entities in the framework:

- CAS Server: The purpose of the CAS server is to issue signed authorization assertions to the users in the community. In the current model, authorization assertions are explicit access rights in the form of an <action, target> tuple. For example, the tuple <read, directory on specific ftp server> represents read permission on a specific directory on an FTP server in the community. However, in an RBAC model, the authorization statement should be the role membership of the user. Hence, the <action, target> tuple takes the form <role in the community,

resources in the community>. For example, an authorization statement <Project Manager, Community A> issued to a user (say, u_1) specifies that u_1 has membership of the role of Project Manager in the community "A". However, the access rights for u_1 largely vary across domains based on the mapping of domain's local role to the Project Manager role in the community and the access rights associated with such a mapped local role.

- Resources: CAS-enabled resources are responsible for parsing the CAS credentials issued by a CAS server and enforcing it on the user. In the RBAC model, CAS credentials represent the community role membership of the user. Hence, the resources need to have knowledge of the community roles created by the CAS administrator and their mapping with the local roles. This can be achieved through an informal communication between the community administrator and the local domain administrator. Thus, the resources need to parse the role and match it with the list of roles in the community and their corresponding mapped roles in the local domain. The users are authorized for access rights associated with the mapped local roles.

- Users: Users will be given role membership in the community rather than explicit access rights on the resources in the community.

5.1.3 CAS Configuration with Integrated Policy

The proposed policy integration tool generates two X-RBAC policy components, namely, XRS and XURAS. An XRS specifies a role's attributes and its hierarchical level that can

be used in configuring the CAS database, whereas an XURAS can be used by the community administrator to verify the user's attributes before assigning him or her membership of the role in the community. Figure 27 details the configuration procedure of the CAS server. The access rights issued to a community member in the RBAC model is his or her role membership, and hence, each action object is configured to represent a role in the community. Thus, all the actions can be stored into a single service type representing a single integrated global access control policy for the community. As the access control policies of the resources have been merged already, all the resources in the community can be configured to represent a single resource group or object group. While granting the permission, each user group is granted with the membership of a single role (or action object) to community resources (or object group).

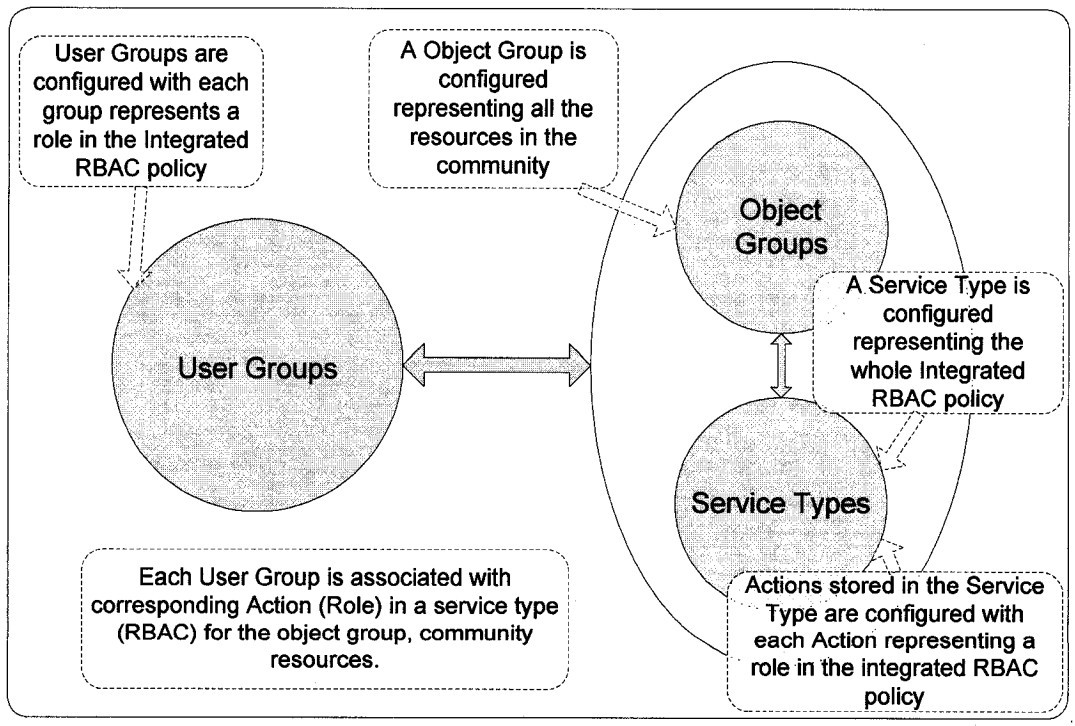


Figure 27. Configuration Details of CAS Server Access Control Policy

The following set of commands can be used to configure the CAS database for a community "A" with the RBAC policy.

admin% cas-enroll –c cas-service-uri object administrator-group communityA-resource namespace

This command adds the communityA-resource as an object and gives all members of the administrator group rights to manipulate the object.

admin% cas-group-admin –c cas-service-uri object create administrator-group communityA-resources

This creates an object group called communityA-resources (representing all the resources in community A), and the members of administrator-group get all rights to this group.

admin% cas-group-add-entry –c cas-service-uri object communityA-resources namespace communityA-resource

The above command adds the communityA-resource object into communityA-resources object group.

admin% cas-enroll –c cas-service-uri serviceType administrator-group IntegratedRBAC

This command creates a serviceType called "IntegratedRBAC".

admin% cas-action –c cas-service-uri add IntegratedRBAC Role

This command adds the community RBAC role into the serviceType Integrated RBAC.

admin% cas-group-admin –c cas-service-uri user create administrator-group Role-in-A

This will create a user group Role-in-A (user group representing a role in community A) and gives all the members of the administrator group the permission to manage the group (i.e., add users, remove users, and so on).

admin% cas-rights-admin –c cas-service-uri Role-in-A objectGroup community-A serviceAction IntegratedRBAC Role

cas-service-uri: URI for the CAS web service
administrator-group: Name of the user group who has all the permissions on service-type object
namespace: Namespace associated with the resource

This command associates the user group with the serviceType and the Object group.

5.2 Web Service as a Resource in CAS

Web services are an emerging technology in multi-domain environments, making them potential resources in collaborations. However, in a community-based authorization framework, at present only the GSIFTP server provided by the Globus Toolkit supports CAS. Hence, this section of the chapter presents extensions to web services that allow them to used as resources with the CAS server. During a web service invocation the client program should pass the user proxy certificate to the web server. The server will have to verify the CAS server signature for SAML assertions and parse the assertion to find the role membership of the user.

5.2.1 Passing Certificate from the Client to the Server

In PKI, one of the applications of the certificates is to digitally sign the messages between two entities to preserve message integrity. A PKI-based digital signature scheme relies on public key cryptography where a user will have a pair of keys: a public key and a private key. A public key is distributed freely by embedding it in the user's digital identity certificate issued by a certificate authority. In a digital signature, the sender digitally signs the messages with his private key and the receiver of the message verifies the digital signature with the sender's public key. Digital signatures are commonly used in client-server communications, wherein the server verifies the digital signature using the

client's certificate. However, storing the certificates of each client is impractical. Hence, a direct reference method is used in many systems whereby the sender of the message attaches his certificate along with the message. Such a direct reference method of certificate passing can be effectively used in sending the client proxy certificate to the server in a community environment, solving two problems – namely, digital signature and proxy certificate passing.

WSS4J (Web Service Security for Java) is a Java library from OASIS Web Services Security TC[3] that can be used to sign and verify SOAP messages with WS-Security information. WSS4J provides special handlers that can be added to the service deployment descriptor (wsdd file) to add a WS-Security layer to an Axis-based web service [61]. WSS4J makes extensive use of *keytool* for retrieving the private keys and the certificates of a user. *Keytool* is a key and certificate management utility that manages a keystore (database) of private keys and their associated X.509 certificate chains for authenticating the public keys. The signing of the SOAP messages is done using special handlers provided by WSS4J as shown in Figure 28.

[3] Source: *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss,* accessed on Aug, 2006.

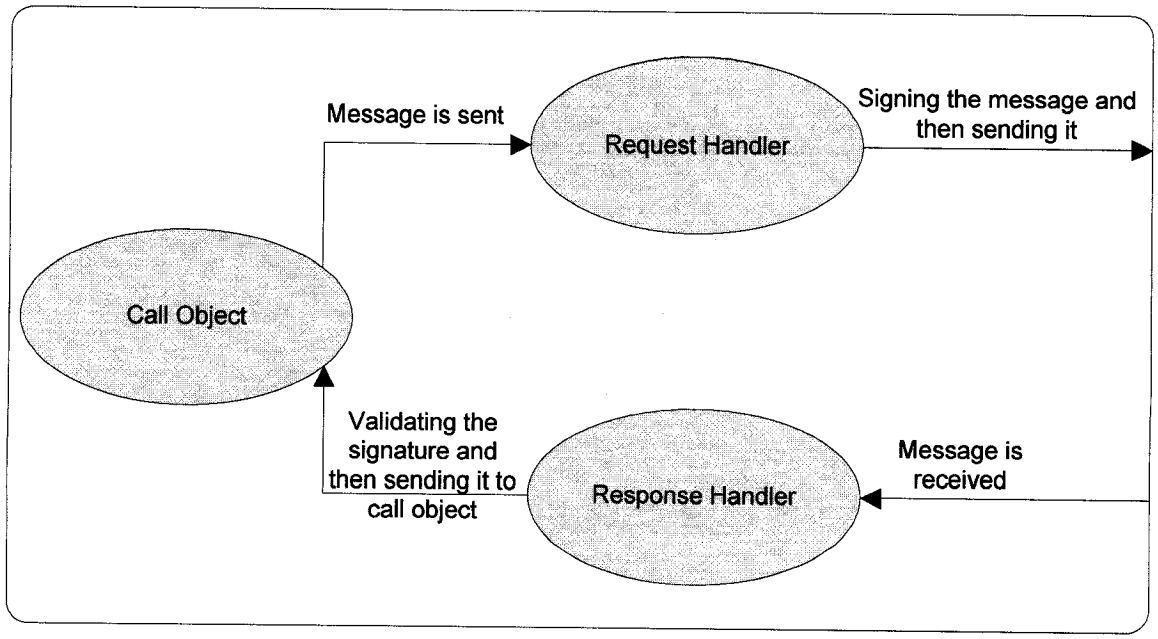


Figure 28. WSS4J Handlers for Signing the SOAP messages

5.2.2 RBAC model for Web Services

Web services participating in collaboration with the CAS authorization framework should be enabled to enforce an integrated access control policy. A user prior to accessing a web services should contact the CAS server to request the CAS credential, which is a role membership of the user in the collaboration. This credential is signed by the CAS server. In the previous section, a method of passing a CAS credential along with a SOAP request message was presented. Upon receiving the credential, a web service should perform the following steps:

- Validate the signature and time period of the CAS credential.
- Enforce the access rights of the role specified in the CAS credential. In a web service context, the access right associated with the role to invoke the requested method is authorized.

- Optionally, the local domain can enforce additional restrictions on the user, based on their local policies.

5.3 Summary

In this chapter extensions to the community authorization framework were presented. The integration of the proposed policy-merging tool with the CAS server enhances the server functionality in the formation of the community. This extension reduces the administrative overhead in establishing a CAS policy. However, such integration introduces the requirement for use of RBAC model among the community resources. Web services are an emerging technology for inter-domain service access and a potential resource in collaboration. Hence, an RBAC model for web-services-as-resources-in-a-community model is presented as the second extension to the community-based authorization framework.

Chapter 6 Validation

In this chapter, the extension to the community-based authorization framework with the proposed policy integration mechanism is validated using a case study implementation. The scenario implemented is the aggregation of data from different web services belonging to different security domains with separate local access control policies.

In the testing phase, the proposed policy integration mechanism is tested for two principles, which should be enforced by an access control framework in a multi-domain environment.

- Security Principle: If an access is not permitted within an individual system, it must not be permitted under secure interoperation.

- Autonomy Principle: If an access is permitted within an individual system, it must also be permitted under secure interoperation.

Also, web server performance results were taken to measure the overhead of the RBAC model in web services.

6.1 Case Study

6.1.1 Scenario

This section illustrates the proposed policy integration tool and the CAS extensions by considering collaboration among various web services for the collection of data from the

sensors. The scenario described in this section is implemented as a part of the Sensor Network project. The web services in the collaboration are an aggregation web service, a mapping web service, and a sensor web service. The sensor web service invokes various methods of the sensor and the aggregation web service collects and composes the data. All three services are offered by different organizations. The integration of access control policies facilitates composing different services, inter-operation among these services, and single sign-on (SSO) for the user to access different services in separate security domains.

Figure 29 below shows the whole process of authorization in a community-based authorization framework. There are three security domains – namely, Domain A, Domain B, and Domain C – involved in the collaboration. The process of authorization is described below:

1. Each domain in the collaboration exports XRS and XURAS components of the local policies.

2. The domain policies are integrated using a user-credential based integration mechanism and applied onto the CAS server. The CAS administrator communicates with the individual domains about the roles in the integrated policy and their mappings with the respective local domain roles.

3. A user in the community initiates a request for a proxy certificate from the CAS server.

4. The CAS server authenticates the user and replies back with SAML authorization assertions.

5. The user initiates a request to access the resources in the community by passing his or her proxy certificate with the SAML assertions issued by the CAS server.

6. The resource verifies the proxy certificate and enforces the community policy for the user on the resources by mapping the community role to the local domain role.

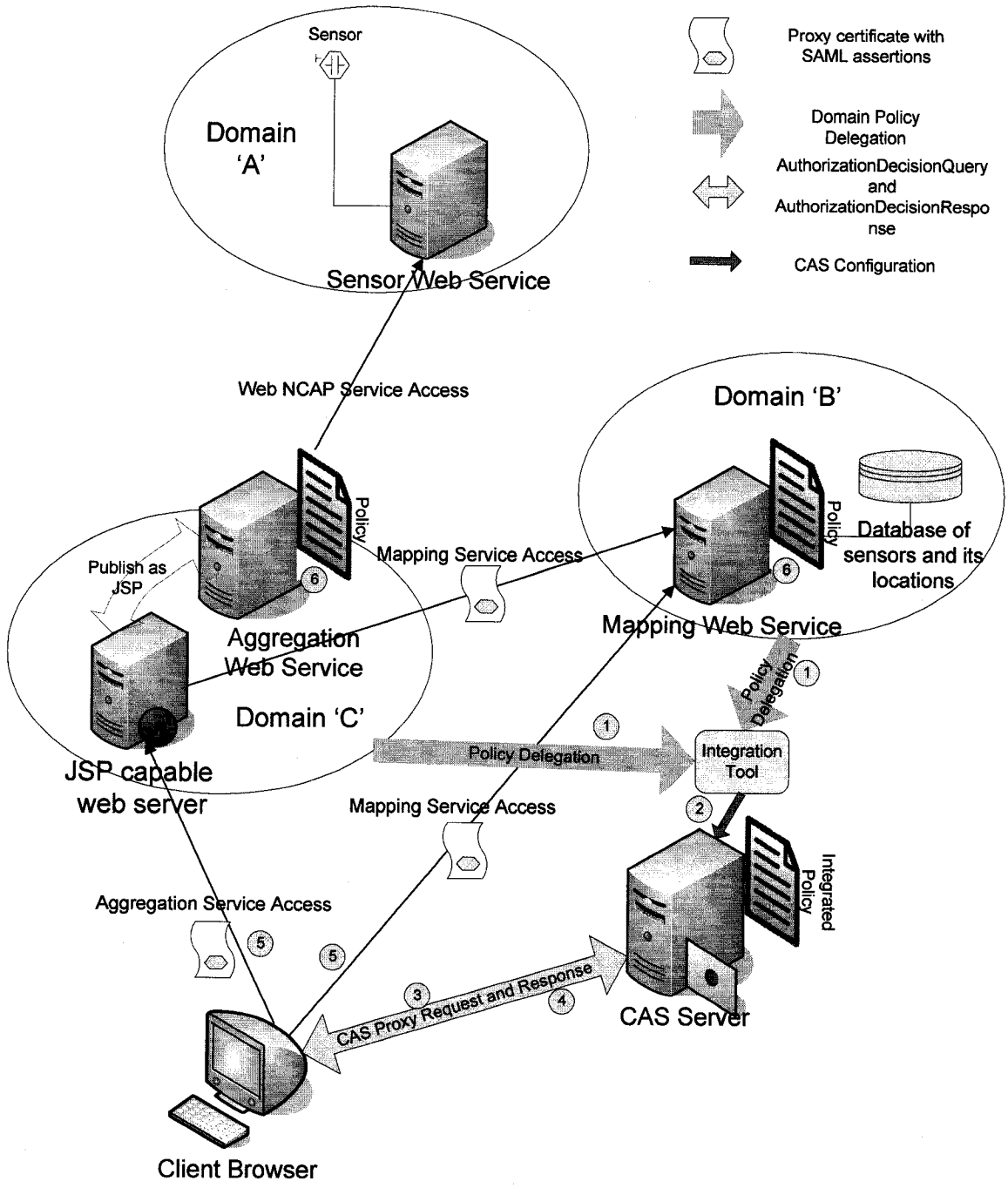


Figure 29. Case Study

Table 5 describes various methods available for different web services (or resources). A mapping web service maintains a database of sensors and their locations. A sensor web service is a 1451.1 NCAP-enabled web service, termed "Web NCAP," that provides

methods to get the readings from a sensor [53]. An aggregation web service composes the data collected from the sensor web service. The aggregation web service client is implemented on a JSP-enabled web server to integrate the aggregation web service interface with location-based visualization services provided by *Google Map* APIs. The JSP-enabled web server forwards the client proxy certificate to the aggregation web service and the mapping web service on behalf of the user.

Web Service	Method	Description
Mapping Service	Add Mapping	Adds a sensor and its location into the database
Mapping Service	Modify Mapping	Modifies the location of the sensor in the database
Mapping Service	Show Sensors	Shows all the sensors and their locations
Mapping Service	Find Nearest Sensor	For a given longitude and latitude, gives the nearest sensor present within 300 miles
Sensor Service	Sensor Read	Gives the measured temperature, humidity, air flow, light etc from a sensor
Sensor Service	Close Sensor	Stops the sensor readings
Aggregate Service	Calculate Average Temperature	Gives average temperature over a period of 5 seconds by invoking Sensor Read method of Sensor Web Service
Aggregate Service	Shutdown Sensor	Shuts down the sensor by invoking Close Sensor method of Sensor Web Service

Table 5. Different Types of Services in the Collaboration

The configuration details used for running different web services (both server and client) is given in the below table.

Host	OS	Platform	Services	Description
137.122.91.190	Fedora Core	Eclipse 3.2.1 Integrated Development Environment (IDE)	Community Authorization Service	**CAS Server** • Globus toolkit version 4.1.1 integrated with CAS service is deployed • PostgreSQL is configured to be used as the backend database • Simple CA is setup with the Nick name 'Globus SimpleCA'.
137.122.91.19	Windows	Eclipse 3.2.1 IDE	Mapping Web service and Aggregation Web service	**Aggregation and Mapping server** • Tomcat 5.5 application server is deployed • Apache Axis 1.2 SOAP server is installed over Tomcat server.
137.122.91.31	Windows	Netbeans IDE	Sensor Web Service	**Sensor Server** • Java application server
137.122.91.19	Windows	Eclipse 3.2.1 IDE	Mapping Web service client and Aggregation Web service client	**Aggregation and Mapping client** • Tomcat 5.5 application server is deployed • Apache Axis 1.2 SOAP server is setup over Tomcat server.

Table 6. Configuration Details

6.1.2 Participating Domain Policies

Figure 30 depicts the graphical annotation of the Domain B and Domain C RBAC policies participating in the collaboration. Domain B has the following roles: Administrator, Employee, and User. The administrator role inherits all the permissions of the roles Employee and User whereas Employee inherits all the permissions of the role User. Domain C has the following roles: Project Administrator and User. The project administrator role inherits all the permissions of the role User in Domain C. Both of the RBAC policies are represented using the X-RBAC model before integration. Table 7 lists

the roles, permission authorizations, and the user credentials associated with each role in both of the domains. Permission authorization defines the access rights (i.e. method invocation in a web service context) available to the corresponding roles in the local domain.

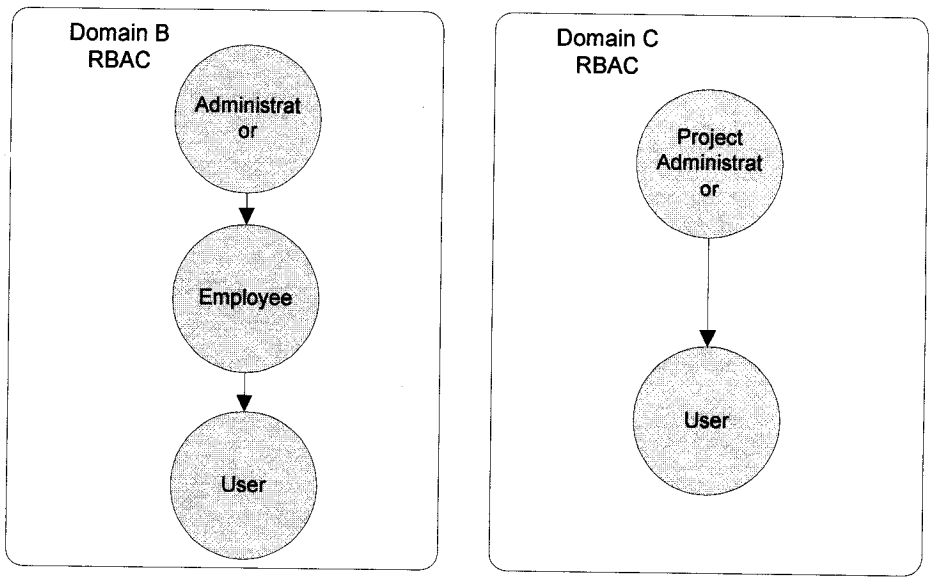


Figure 30. RBAC Policy Graph Models for Domains B and C

Domain	Role	Permission Authorization	User Credentials
Domain B	Administrator	'Add Mapping' and 'Modify Mapping' method invocation on mapping web service	Employee at Domain B and Position is Manager
Domain B	Employee	'Show Sensors' method invocation on mapping web service	Employee at Domain A
Domain B	User	'Find Nearest Sensor' method invocation on mapping web service	Any User (No Credentials)
Domain C	Project Administrator	'Shutdown Sensor' method invocation on aggregate web service	Employee at Domain A and Position is Project Administrator
Domain C	User	'Calculate Average Temperature' method invocation on aggregate web service	Any User (No Credentials)

Table 7. Description of Roles Involved in the Collaboration

6.1.3 Policy Integration Results

Figure 31(a) shows the display of the community RBAC policy generated using the user-credential based integration tool. Figure 31(b) depicts the RBAC policy graph of the generated policy showing the roles from both the domains, Domain B and Domain C, in one graph. The integration tool has generated two types of inter-domain mappings, namely, equivalence role mapping and contained role mapping. User roles from both the domains are merged together, and hence users in both the domains get the access rights of the role User in the collaborating domain. The Project Administrator in Domain C is a parental role to the Employee in Domain B in the integrated policy, which implies that a user, u1, authorized for the Project Administrator role in Domain C can inherit the permissions of the foreign role - Employee of Domain B.

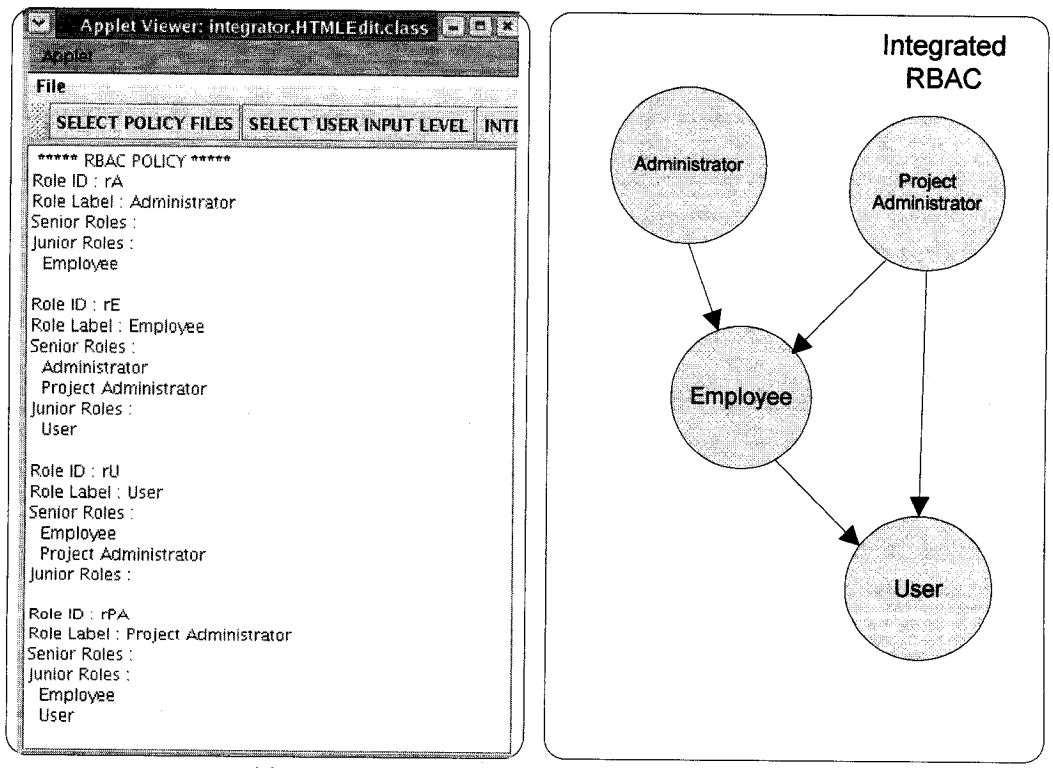


Figure 31. RBAC Policy Integration Results

6.1.4 Access Control Testing

For the basis of the access control test, a community administrator (named *Globus*) is created with the administrative permission on the CAS server by creating a Linux user account and updating the configuration file, *casDbProperties*. Four members *Alice*, *Bob*, *Carol*, and *Dave* are assigned different roles in the community by *Globus*. The roles assigned to each of the users are as follows: Alice, membership to the Administrator role; Bob, membership to the Project Administrator role; Carol, membership to the Employee role; and Dave, membership to the User role. The CAS proxy certificates issued to the members are made valid for a year and contain the signed SAML assertion specifying the role membership of the client. Once the members obtain the proxy certificates, these certificates are stored in a keystore utility tool at the aggregation web service client with a unique password for each of the user certificate in the keystore. Hence, to invoke any web service methods, the user needs to enter his or her username and password to access the corresponding CAS certificate from the keystore and pass it on with the SOAP request message. Table 8 shows the test results for each of the users for access restrictions on the available services in the collaboration.

Test Case	Role	User	Access Rights	Expected Result	Actual Result
1	Administrator	Alice	Shutdown-sensor	Deny	Denied
2	Project Administrator	Bob	Add-mapping, Modify-mapping	Deny	Denied
3	Employee	Carol	Add-mapping, Modify-mapping, and Shutdown-sensor	Deny	Denied
4	User	Dave	Add-mapping, Modify-mapping, Show-sensor, and Shutdown-sensor	Deny	Denied

Table 8. Testing for Access Restrictions

Also, a complete list of permission and restrictions for each user for different services are given in the following table.

Service	Alice-Administrator	Bob-Project Administrator	Carol-Employee	Dave-User
Add-mapping	√	X	X	X
Modify-mapping	√	X	X	X
Show-sensor	√	√	√	X
Find-nearest-sensor	√	√	√	√
[illegible]	X	√	X	X
[illegible]	√	√	√	√

Table 9. Test Results (X– Method Not Accessible, √- Method Accessible)

Figure 32 shows the JSP interface of the aggregation web service client program. The JSP interface is implemented as part of the sensor network project. Dave has logged into the system and has accessed the average temperature near his region service. This is a composite service implemented using all three web services. The mapping web service is called to find the nearest sensor located in the region selected by Dave on the Google map. Then the aggregation web service is invoked to calculate the average temperature that, in turn, invokes the sensor web service each second over a period of five seconds. Out of the two markers in Figure 32, one indicates the region selected by Dave and the other indicates the location of the actual sensor deployed and the average temperature measured.

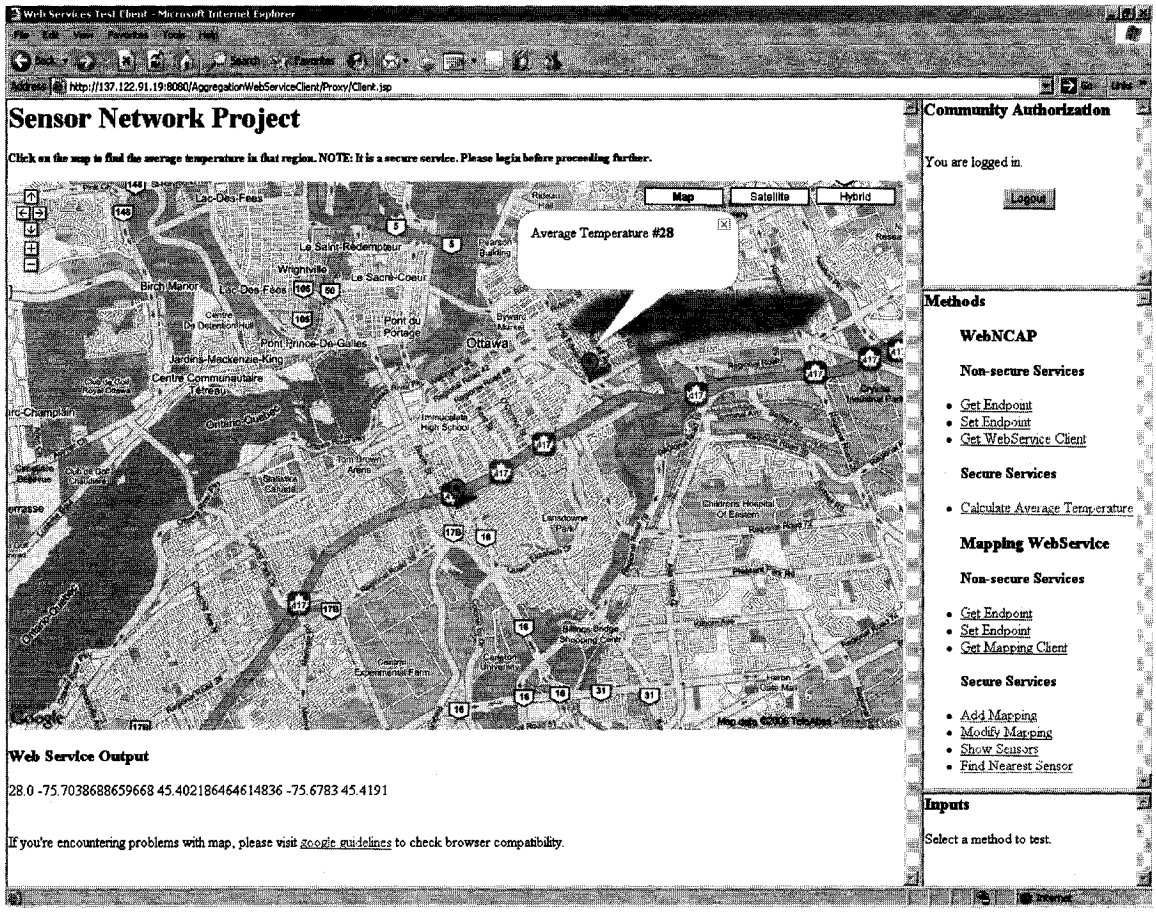


Figure 32. User Interface to the Aggregation Web Service Client

The details of the implementation of the case study can be found in Appendix B.

6.2 Testing the System

6.2.1 Verification of the Integrated Policy

Each time a new access control policy is created for the collaboration, it has to be tested for two principles, namely, the security principle and autonomy principle.

- Security Principle: If a user is not permitted within a local domain, then he or she must not be permitted under collaboration.

- Autonomy Principle: If a user is permitted within a local domain, then he or she must also be permitted under collaboration.

In this section, the proposed policy integration model is analyzed for conformance with respect to the two defined principles above using four complex security policies, shown in Figure 33. The principles define the criteria for verifying policy correctness. The process of access control policy integration during the collaboration may modify the participating domain policies. But such modifications should not change the access privileges of the local users or else should be within acceptable limits.

Different test results can be obtained by integrating any two policies out of the available four test policies, shown in Figure 33, in different combinations. In Figure 33, the nodes represent the roles and the edges represent the hierarchical inheritance between the roles. Each role is uniquely labelled. Based on the user-credential set associated with these roles, different kinds of mappings are possible. The following is the list of possible mappings between the four presented RBAC policies:

- Roles R11, R21, R31, and R41 are equivalent
- Roles R16, R210, R37, and R43 are equivalent
- Roles R18, R211, R36, and R46 are equivalent
- Roles R15 and R33 are equivalent
- Roles R12 and R22 are equivalent
- Roles R24, R34, and R47 are equivalent

- Role R29 is contained in R15, R13
- Role R32 is contained in R12, R22
- Role R44 is contained in R12, R22
- Role R17 is contained in R24, R34, and R47

In Figure 33, the user-credential set associated with the equivalent roles are indicated by the same pattern for better understanding.

For instance, consider the integration of Policies 1 and 2 using the user-credential based role-mapping algorithm described in Tables 3 and 4 in Chapter 4. The algorithm performs role integration by examining each of the roles in RBAC Policies 1 and 2 for possible inter-domain mapping. The roles in the RBAC policies are examined in top-to-bottom fashion. The inter-domain roles R11 and R21 are equivalent just as the user credentials associated with both the roles are equivalent. Hence, a new role – say, Ra – is created in the integrated RBAC policy with an associated user-credential set $UC_{set}(R11)$ (since $UC_{set}(R11) = UC_{set}(R21)$). Similarly, roles R16 - R210, R18 - R211, and R12 – R22 are equivalent based on their associated user-credential set. New roles – Rb, Rc, and Rd – are created in the integrated policy for each of the above-identified sets of equivalent roles. Another type of mapping found between the roles R17 and R24 is containment mapping. The user-credential set $UC_{set}(R17)$ is contained in $UC_{set}(R24)$. To create the containment mapping in the integrated policy, both role R17 and R24 are created with a link between them, making R17 a parent role to the R24 role. The remaining unmapped roles from Policy 1 and Policy 2 are created in the integrated policy with all the links between them

as defined in the individual policies 1 and 2. Figure 34 depicts step by step process of integrated policy generation before the verification process. Once integrated, the integrated policy is verified for any conflicts.

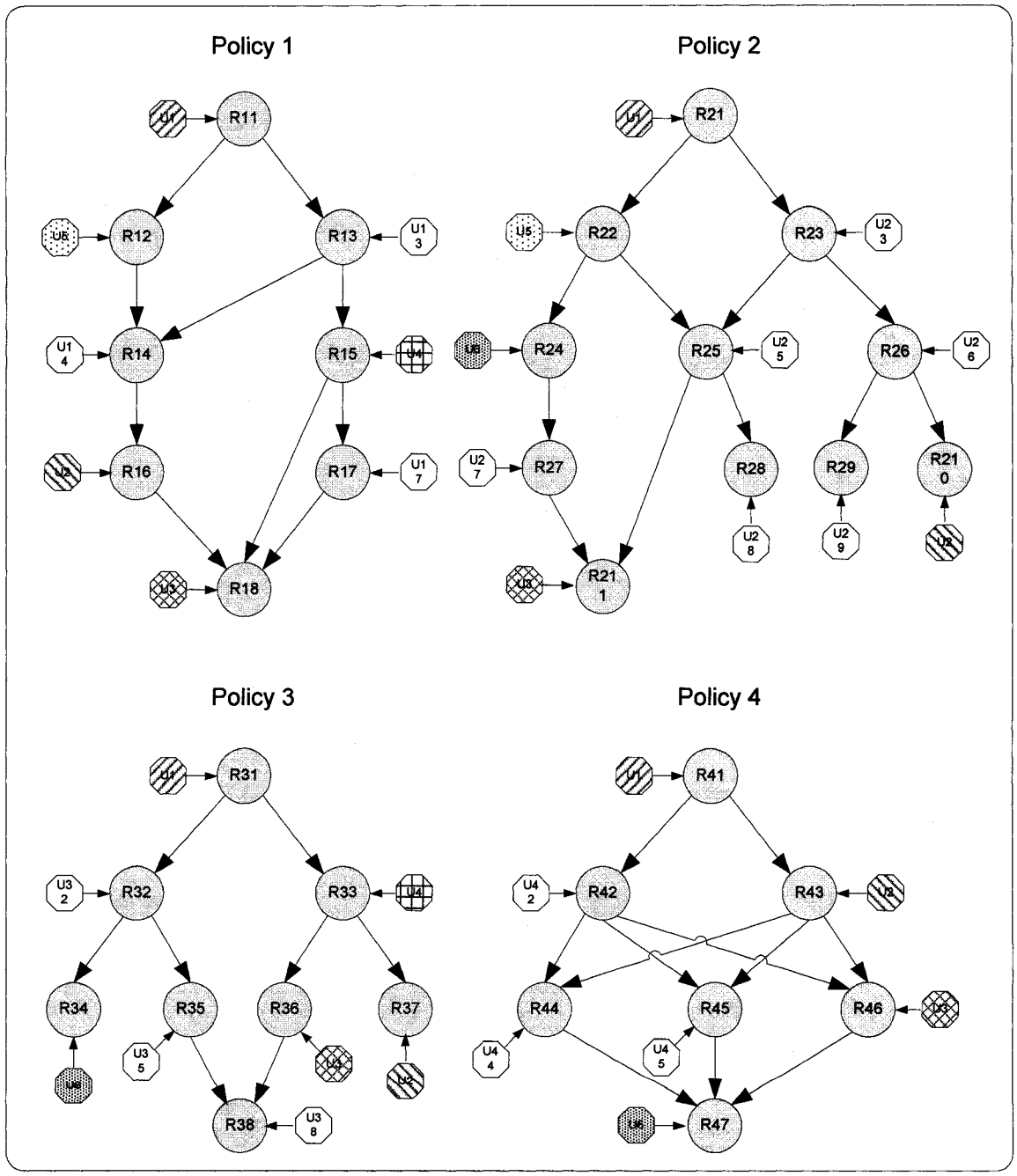


Figure 33. RBAC Policies for Testing the System

The inter-domain link created between the role R26 and Rc (the new role created from the merging of roles R16 and R210) has deduced a role mapping between roles R210 and R211. Roles R210 and R211 are not linked in the local policy 2. So, the inter-domain link R26 and Rc can be deleted based on the administrator input. Hence, roles R16 and R210 cannot be merged into a single role Rc.

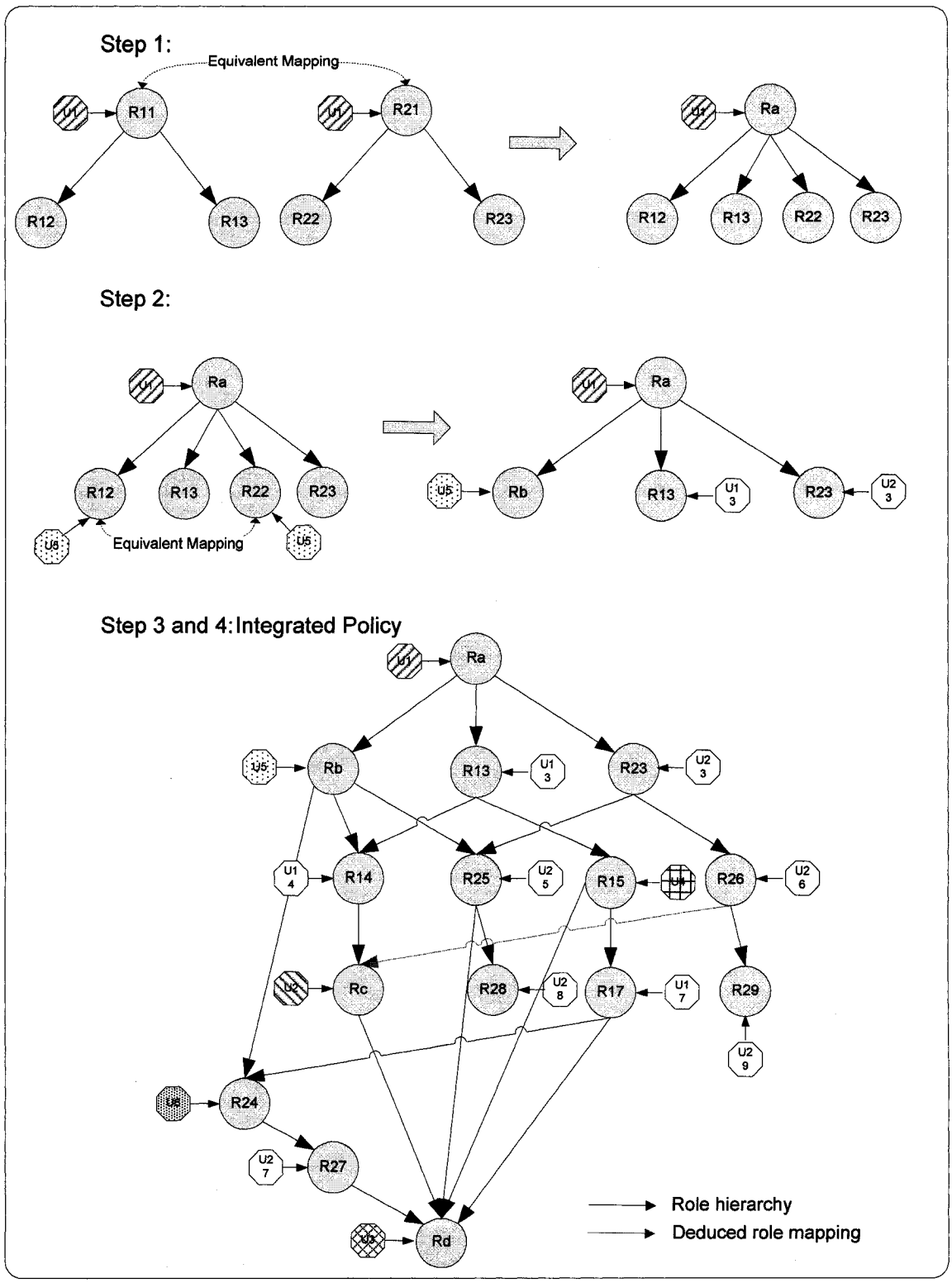


Figure 34. Integrated Policy from Policies 1 and 2

6.2.1.1 Security Principle Conformance

The security principle, stipulating access restrictions, is validated in this section, by formally defining the upper bound and lower bound for the increase in the accessibility due to policy integration process, as shown in Figure 34. As discussed in Section 4.2.3.3, accessibility increase in a participating domain is due to newly created inter-domain role mappings during the integration process. At any inter-operation, the multiple values for accessibility are due to different possible inputs by the security administrator. A value for the increase in accessibility can be computed using the following formula:

Accessibility Increase = Ratio of (Deduced role mappings) to (Total number of possible role mappings)

From the test results, the variance of the upper bound is calculated to be $1.204 * 10^{-4}$.

However, the ideal value is always zero at any time, meaning that there are no deduced role mappings due to policy integration, which is equal to the calculated lower bound. The lower bound is always zero because the security administrator can remove all such deduced role mappings during the integrated policy verification process. This indicates that the integration mechanism is fully in conformance with the security principle.

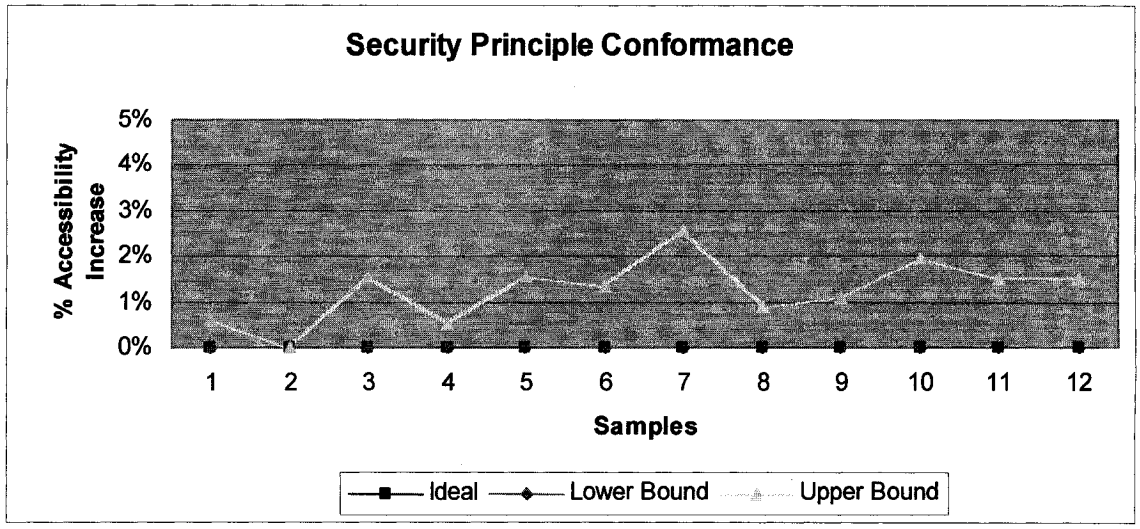


Figure 35. Security Principle Conformance

6.2.1.2 Autonomy Principle Conformance

The proposed RBAC policy integration process does not remove any of the existing links among roles of a participating domain policy. Thus, all the relations between roles specified in the collaborating domains are implied in the integrated policy. At any inter-operation, the decrease in accessibility is computed using the following formula:

Accessibility Decrease = Ratio of (Number of mappings removed during integration) to (Total number of existing role mappings)

Since none of the existing intra-role mappings were removed during the policy integration, the value is always the ideal value, which indicates that the integration mechanism is fully in conformance with the autonomy principle.

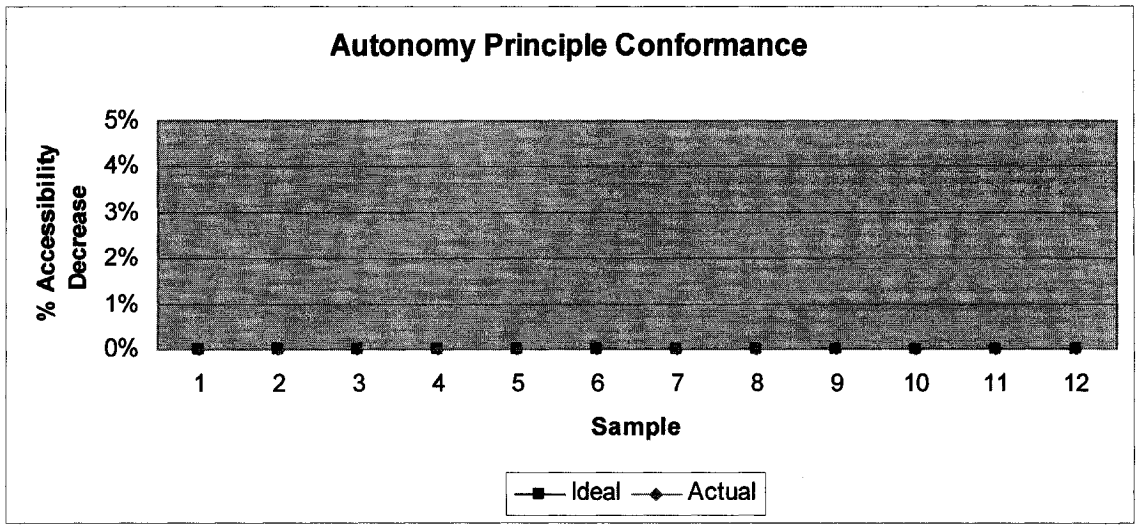


Figure 36. Autonomy Principle Conformance

6.2.2 Performance Testing

The experiments were conducted to measure the overheads of the RBAC model for a web service. The machine used for the experiments was configured with a Pentium IV-2.19 GHz processor, 512MB of RAM, and Windows Professional 2002. A Tomcat application server 5.5 and an Apache axis 1.2 SOAP engine were installed on the machine to host the web services. The following factors affected the web service response time:

- Verification of a client CAS proxy certificate attached with the SOAP request.
- Extraction of SAML assertions embedded in the proxy certificate.
- Mapping the community role membership of the client to the local roles.
- Authorizing the client-invoked web service method against the local role.

A TCP/IP port monitor feature in Eclipse IDE 3.2.1 was used to capture the sequence of messages between the server and client to measure the server response time.

In the first set of experiments, the response time of the Tomcat application server was measured for the following three cases:

- Web service without a RBAC model and a client SOAP request without a CAS proxy attachment.
- Web service without a RBAC model, but the client SOAP request is digitally signed using the CAS proxy certificate. The SOAP request has a direct reference to the client CAS proxy certificate as an attachment to the request.
- Web service with a RBAC model to authorize the client request and the client SOAP request is digitally signed and has a direct reference to the CAS proxy certificate.

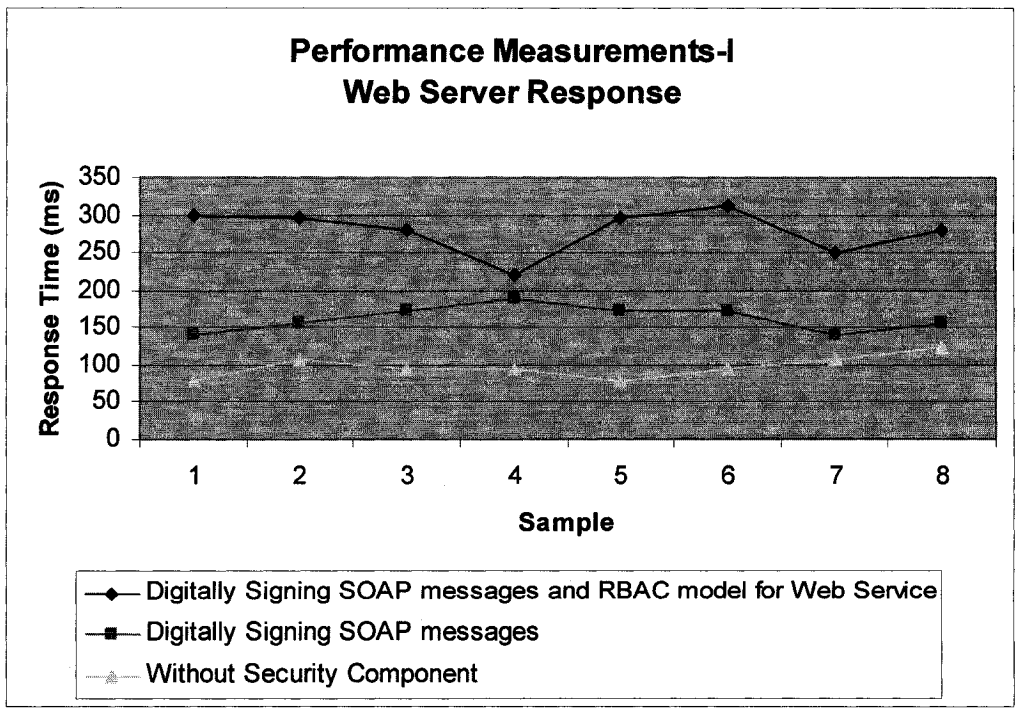


Figure 37. Performance Measurements-I

The test results show the overhead in the response time for using the RBAC model and digital signing of SOAP messages in a Web service. Note that the above test results do not include the SOAP message delays. So the overhead is due to the processing required in digital signature verification and authorization of the user in the RBAC model.

In the second set of experiments, the web service performance in terms of response time was measured for two different methods of client CAS proxy reference at the server side, namely, direct reference in the SOAP message and certificate store reference. In direct reference, the client attaches the certificate along with a SOAP request message, whereas in certificate store reference, the client specifies the distinguished name of the issuer of the certificate used for signature as well as the serial number of the certificate. Client certificates are usually stored in a database using a Keytool utility. Keytool is a key and certificate management utility that manages a keystore (database) of private keys and their associated X.509 certificate chains for authenticating the public keys.

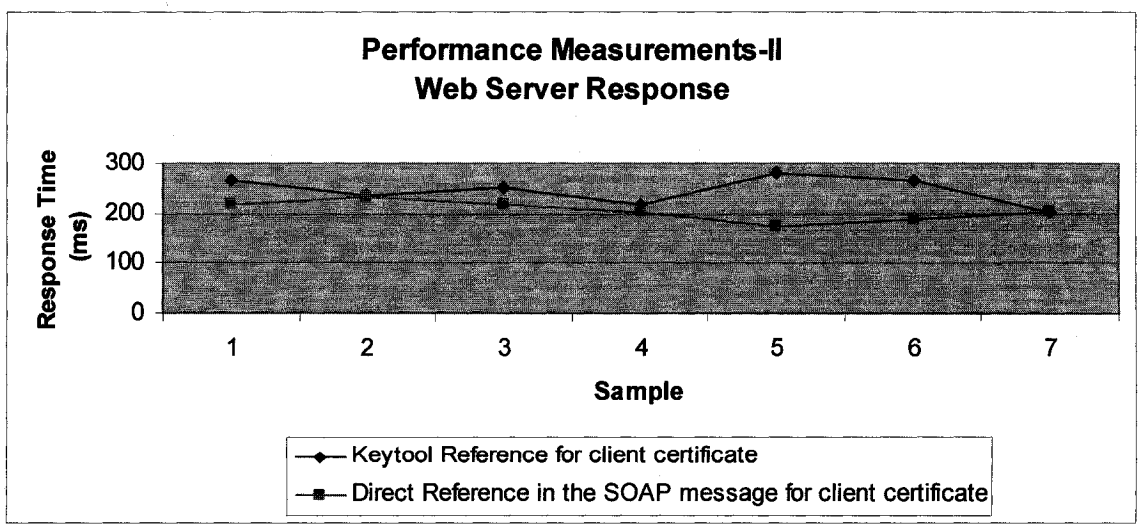


Figure 38. Performance Measurements-II

The test results show the difference in the web server response time due to the use of different techniques for the SOAP digital signature validation process.

6.3 Summary

In this chapter, a case study of the collaboration of web services with local access control policies was presented with prototype implementation. Also, the policy integration mechanism was tested for autonomy and security principles. The results show that the user-credential based integration technique is in conformance with the principles. The preliminary web service performance results were taken to measure the overhead of the X-RBAC implementation for the web services.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

In this thesis, the problem of integrating access control policies from different heterogeneous domains is addressed in order to facilitate the bootstrapping of collaboration in a convenient and cost-effective manner.

The central idea of the thesis is to use user-credential based role mapping to merge the policies in order to minimize the amount of administrative overhead and facilitate the integration of autonomous access control policies into heterogeneous environments. The participating domains have to define and document the attributes; the users have to take up the roles. Hence, inter-domain role mappings are established based on the matching

results of the user-credentials associated with the roles. In such a collaboration, a user can access resources in foreign domains by inter-domain role mappings. However, a user's access will be based on his or her qualifications in the local domain.

The requirements for a RBAC policy integration technique identified in Chapter 3 are discussed below to confirm that they are met by the research work:

- Assignment of integrated RBAC roles to the users: The proposed RBAC policy integration technique generates XURAS and XRS-XML sheets that contain the user qualifications required by the users in the collaboration to take up particular roles in the integrated RBAC. Hence, a TTP administrator can assign new users to the roles based on the conditions specified in XURAS.

- Participant Privacy Policy: Most of the proposed policy integration models are based significantly on the access rights to the resources that might breach the privacy policy of the domains during collaboration. In the user-credential based role mapping, no assumptions are made on the access rights and resources, thus placing no restrictions on resource homogeneity and the domain privacy policy.

- Environment Heterogeneity: In the collaboration, the diversity is vast among the participating resources and the services rather than among the users accessing these services. The user-credential based role mapping mechanism can effectively integrate the policies even in a heterogeneous environment. In a case study in Chapter 6, diverse web services were effectively integrated using the proposed policy integration model. The web services considered in the case study had different object and action sets, and hence, it would not have been feasible to

integrate the web service policies with the integration models based on access rights.

Thus with the use of a user-credential based role mapping technique to integrate the security policies, all the requirements defined in Chapter 4 have been met.

In this thesis, the community authorization model is extended to include the proposed policy integration mechanism and to support the RBAC model in a community. The community model is an authorization framework for security in federated systems. Such an integration of the proposed policy composition tool into a community model will enhance its functionality by eliminating the inconsistencies in the policies of the participating domains. The extension also includes the addition of a security component into the web service model to make the model CAS-enabled resources in the community.

The integrated policy should be consistent and conflict-free with participating domains, otherwise, the entire integration process fails. Therefore, the proposed model is tested using security and autonomy principles to verify the integrated global policy against the participating domain policies.

7.2 Future Work

The research work reported in this thesis can be extended into several directions as summarized below.

- *Semantic-based user-credential matching for inter-domain role mapping*: Conflict resolution in user-credentials matching in the proposed integration mechanism is

indispensable since the individual policies are written independently. The current approach of synonym-based matching with the use of regular expression only ensures that naming conflicts and expression conflicts are resolved. This conflict resolution mechanism cannot guarantee correct matching because similar user-credentials can be represented differently. In such cases, semantic-based user-credential matching [25, 26, and 41] could be a more efficient matching mechanism in the proposed integration system.

- *Constrained RBAC model in a community framework*: The underlying assumption while designing integration mechanisms is that the participating domains implement a hierarchical RBAC model. But in a hierarchical RBAC model, it is possible to define various constraints, such as user-specific separation of duty, role-specific separation of duty, and temporal constraints. Therefore, the proposed integration mechanism can be extended in the future to incorporate a constraint-based RBAC model.

- *Enforcement of community policies along with an integrated global policy*: Collaborations keep evolving in an ever-changing business world. Enforcing new restrictions on integrated policies is not a new phenomenon. However, due to the use of an RBAC model in the community-based authorization framework, the resources can only enforce the permission authorizations associated with the local roles that are mapped with the community role. Therefore, the current CAS-enabled PEP available in the Globus FTP server needs to be upgraded to support the RBAC model and to support web services in the community.

References

[1] Akenti – A Distributed Access Control System, Available from http://www-itg.lbl.gov/Akenti, 2006.

[2] Al-Kahtani, M. A., Sandhu, R., "A Model for Attribute-Based User-Role Assignment," Proceedings of the 18th Annual Computer Security Applications Conference, 2002.

[3] Bhatti, R., X-GTRBAC: An XML-based Policy Specification Framework and Architecture for Enterprise-Wide Access Control, Master Thesis, Purdue University, Available from Purdue University as Technical Report, 2003.

[4] Bhatti, R., Joshi, J., Bertino, E., Ghafoor, A., "Access control in dynamic XML-based Web-services with X-RBAC," Proceedings of The First International Conference on Web Services, 2003.

[5] Bidan, C., and Issarny, V., "Dealing with Multi-Policy Security in Large Open Distributed Systems," INRIA, 1997.

[6] Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A., "The KeyNote Trust Management System," RFC 2704, IETF, 1999.

[7] Boella, G., Van der Torre, L., "Local policies for the control of virtual communities," In Procs. of IEEE/WIC Web Intelligence Conference, 2003.

[8] Bonatti, P., De Capitani di Vimercati, S., Samarati, P., "An Algebra for Composing Access Control Policies," ACM Transactions on Information and System Security, Vol. 5, No. 1, 2002, pp. 1-35.

[9] Butler, R. et al., "Design and Deployment of a National-Scale Authentication Infrastructure," IEEE Computer, Vol. 33, No. 12, 1999, pp. 60-66.

[10] Chen, Y., et al., "Design and Implementation of Dynamic-Role Based Access Control Framework in Grid Environment," ITCC, 2005, pp. 758-759.

[11] Coetzee, M., Eloff, J. H. P., "Virtual enterprise access control requirements," Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology, 2003.

[12] Cuppens, F., Cholvy, L., Saurel, C., Carrère, J., "Merging Security Policies: Analysis of a Practical Example," CSFW, 1998, pp. 123-136.

[13] Dunlop, N., Indulska, J., Raymond, K., "Dynamic Policy Model for Large Evolving Enterprises," Proceedings of the 5th IEEE International Enterprise Distributed Object Computing Conference, 2001, pp. 193-197.

[14] Dunlop, N., Indulska, J. and Raymond, K., "Dynamic Conflict Detection in Policy-Based Management Systems," Proceedings 6th IEEE Enterprise Distributed Object Computing Conference, 2002.

[15] Foster, I., "Globus Toolkit Version 4: Software for Service-Oriented Systems," IFIP International Conference on Network and Parallel Computing, Vol. 2, No. 13, 2005.

[16] Foster et al., A Globus Primer or, Everything You Wanted To Know About Globus But Were Afraid to Ask, an Early and Incomplete Draft, Available from http://www.globus.org/toolkit/docs/4.0/key/GT4_Primer_0.6.pdf, 2005.

[17] Foster, I., and Kesselman, C., "Globus: A Toolkit-Based Grid Architecture," The Grid, Blueprint for a Future Computing Infrastructure, Morgan Kaufmann, 1999, pp. 259-278.

[18] Gunnarsson, P., Role based access control in a telecommunications operations and maintenance network, Thesis, Link¨opings universitet, Dublin, Available from http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-2875, 2005.

[19] Gannon, D., Ananthakrishnan, R., Krishnan, S., Govindaraju, M., Ramakrishnan, L., and Slominski, A., "Grid Web Services and Application Factories," Grid Computing: Making the Global Infrastructure a Reality, 2003.

[20] Hamada, T., "Dynamic Role Creation from Role Class Hierarchy – Security Management of Service Session in Dynamic Service Environment," In Proceeding of the TINA '97 – Global Convergence of Telecommunication and Distributed Object Computing, 1997.

[21] Internet2 Middleware Architecture Committee for Education (MACE), Available from http://middleware.internet2.edu/MACE/, 2006.

[22] Jajodia, S., Samarati, P., and Subrahmanian, V. S., "A logical language for expressing authorizations," In IEEE Symposium on Security and Privacy, 1997, pp. 31-42.

[23] Johnston, W., Mudumbai, S., Thompson, M., "Authorization and attribute certificates for widely distributed access control," in Proceedings of the Seventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), 1998, pp. 340–345.

[24] Josang, A., Gollmann, D., Au, A., "A Method for Access Authorisation Through Delegation Networks," Australasian Information Security Workshop, 2006.

[25] Joshi, J., Bhatti, R., Bertino, E., Ghafoor, A., "An Access Control Language for Multidomain Environments," IEEE Internet Computing, 2004.

[26] Kagal, L., Finin, T., Joshi, A., "Trust-based security in pervasive computing environments," IEEE Computer, Vol. 34 No. 12, 2001, pp. 154–157.

[27] Kalfoglou, Y., Schorlemmer, M., "IFmap: an ontology-mapping method based on information-flow theory," Journal of Data Semantics, Vol. 1, No. 1, 2003, pp. 98-127.

[28] Kalfoglou, Y., Schorlemmer, M., "Ontology mapping: the state of the art," The Knowledge Engineering Review, Vol. 18, No. 1, 2003, pp. 1-31.

[29] Kamath, A., Liscano, R., El Saddik, A., "User-Credential Based Role Mapping in Multi-domain Environment," International Conference on Privacy, Security, and Trust (PST), 2006.

[30] Keahey, K., Welch, V., "Fine-Grain Authorization for Resource Management in the Grid Environment," GRID, 2002, pp. 199-206.

[31] Keoh, S.L., Lupu, E., and Sloman, M., "PEACE: A Policy-based Establishment of Ad-hoc Communities," In the Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 20), 2004, pp. 386 - 395.

[32] Kern, A., Walhorn, C., "Rule support for role-based access control," SACMAT, 2005, pp. 130-138.

[33] Khambatti, M., Dasgupta, P., and Ryu, K.D., "A Role-Based Trust Model for Peer-to-Peer Communities and Dynamic Coalitions," Second IEEE International Information Assurance Workshop, 2004.

[34] Koch, M., Mancini, L., Parisi-Presicce, F., "A graph-based formalism for RBAC," ACM Trans. Inf. Syst. Secur, Vol. 5, No. 3, 2002, pp. 332-365.

[35] Li, X., Xu, Z., Liu, X., Yang, N., "Community-Based Model and Access Control for Information Grid," Web Intelligence, 2003, pp. 462-465.

[36] Liu, P. and Chen, Z., "An Access Control Model for Web Services in Business Process," In IEEE/WIC/ACM Int. Conf. on Web Intelligence, 2004, pp. 292-298.

[37] Lorch, M., Kafura, D., "Supporting Secure Ad-hoc User Collaboration in Grid Environments," Proc.3rd Int. Workshop on Grid Computing, 2002, pp 181 – 193.

[38] Lorch, M., and Kafura, D. G., "The prima grid authorization system," J. Grid Comput., Vol. 2, No. 3, 2004, pp. 279-298.

[39] Markus, L. et al., "Conceptual Grid Authorization Framework and Classification," The 7th Global Grid Forum Workshop, 2003.

[40] Minsky, N., Ungureanu, V., "Scalable Regulation of Inter-enterprise Electronic Commerce," WELCOM, 2001, pp. 219-232.

[41] Na. S., and Cheon, S., "Role Delegation in Role-Based Access Control," in Proc. of the 5th ACM workshop on Role-Based Access Control, 2000, pp. 39–44.

[42] Naldurg, P., Roy H., Campbell, M., Mickunas, D., "Developing Dynamic Security Policies," Proceedings of the 2002 DARPA Active Networks Conference and Exposition, 2002.

[43] Nejdl, W., Olmedilla, D., Winslett, M., Zhang, C. C., "Ontology-Based Policy Specification and Management," ESWC, 2005, pp. 290-302.

[44] Noy, N.F., Musen, M., "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," Proceedings of Seventeenth National Conference on Articial Intelligence, 2000, pp 450-455.

[45] OASIS, Security Services TC, Available from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security, 2006.

[46] Osborn, S.L., Sandhu, R., and Munawer, Q., "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," ACM Trans. Information and System Security, Vol. 3, No. 2, 2000, pp. 85-106.

[47] Park, J. S., Hwang, J., "Role-based access control for collaborative enterprise in peer-to-peer computing environments," SACMAT, 2003, pp. 93-99.

[48] Pearlman, L. Welch, V., Foster, I., Kesselman, C., and Tuecke, S., "A community authorization service for group collaboration," In IEEE Workshop on Policies for Distributed Systems and Networks, 2002.

[49] Pearlman, L., Welch, V., Foster, I., Kesselman, C., and Tuecke, S., "A Community Authorization Service for Group Collaboration," IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[50] Pearlman, L., Welch, V., Foster, I., Kesselman, C., and Tuecke, S., "The Community Authorization Service: Status and Future," CHEP03, 2003.

[51] PERMIS – Privilege and Role Management Infrastructure Standards, Available from http://www.permis.org, 2006.

[52] Roure, D., Jennings, D., Shadbolt, N.R., "The Semantic Grid: Past, Present, and Future," Proceedings of the IEEE, Vol. 93, No. 3, 2005, pp. 669-681.

[53] Sadok, E. F., Web Services for IEEE 1451.1 sensor network controllers, MASc Thesis, University of Ottawa, Available from the National Library of Canada, Ottawa, Canada, 2006.

[54] Sandhu, R., Coyne, E., Feinstein, H., Youman, C., "Role-Based Access Control Models," Computer, Vol.29, No.2, 1996, pp. 38-47.

[55] Seleznyov A., Hailes S., "Distributed Knowledge Management for Autonomous Access Control in Computer Networks," Intl. Conf. on Information Technology: Coding and Computing, Vol. 2, 2004, pp. 433-437.

[56] Shafiq, B., Joshi, J., Bertino, E., Ghafoor, A. "Secure Interoperation in a Multi-Domain Environment Employing RBAC Policies," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 11, 2005, pp. 1557 - 1577.

[57] Shehab, M., Bertino, E., and Ghafoor, A., "SERAT : SEcure Role mApping Technique for Decentralized Secure Interoperability," In Proceedings of the ACM Symposium on Access Control, Models and Technologies (SACMAT 05), 2005.

[58] Stoker, G., et al., "Toward Realizable Restricted Delegation in Computational Grids," European High Performance Computing and Networking, 2001.

[59] The Globus Toolkit 4.2 Release, Available from http://www.globus.org/ogsa/releases/alpha/index.html, 2006.

[60] The Shibboleth Project, Available from http://shibboleth.internet2.edu/, 2006.

[61] Tong, K., Developing Web Services with Apache Axis, Ed: First, TipTec Development, Australia, 2005.

[62] Tull, C. E. et al., "Using CAS to Manage Role-Based VO Sub-Groups," CoRR, 2003.

[63] Upstill, C., Surridge, M., "Grid Security: Lessons for Peer-to-Peer Systems," In Proceedings of 3rd IEEE Conference on P2P Computing, 2003.

[64] Van der Vet, P.E., Mars, N.J.I., "Bottom-Up Construction of Ontologies," In IEEE Transactions on Knowledge and Data Engineering, Vol.10, No.4, 1998.

[65] Welch, V., Siebenlist, F., et al., "Security for Grid Services," in Proc. of Twelfth International Symposium on High Performance Distributed Computing (HPDC12), 2003.

[66] Welch, V., Siebenlist, F., Chadwick, D., Meder, S., Pearlman, L., "Use of SAML for OGSA Authorization," GGF, 2003.

[67] Welch, V., et al., "X.509 proxy certificates for dynamic delegation," In 3rd Annual PKI R&D Workshop, 2004.

[68] Welch, V., Barton, T., Keahey, K., and Siebenlist, F., "Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration," Proc. Fourth Ann. Public Key Infrastructure R&D Workshop, 2005.

[69] Yan, G., Ng, W. K., and Lim, E., "Product Schema Integration for Electronic Commerce–A Synonym Comparison Approach," IEEE Trans. Knowledge and Data Eng., Vol. 14, No. 3, 2002, pp. 583-598.

[70] Zhang, L., Ahn, G., Chu, B., "A role-based delegation framework for healthcare information systems," SACMAT, 2002, pp. 125-134.

[71] Zhang, G., Parasher, M., "Dynamic Context-Aware Access Control for Grid Applications," Proc. Fourth Int'l Workshop Grid Computing, 2003, pp. 101-108.

Appendix A:

For the source code of the integration tool and the web service server/client projects, please contact the author at uakamath@gmail.com

The software developed for this thesis utilized the following software and libraries.

Software used:

1. Eclipse IDE version 3.2.1 as the development environment, available at www.eclipse.org
2. Globus Toolkit version 4.0.2 for the CAS server implementation, available at www.globus.org
3. X-GTRBAC version 1.1 as the policy execution engine at the web service server side, available at http://web.ics.purdue.edu/~bhattir/academics/research/projects/x-access.htm

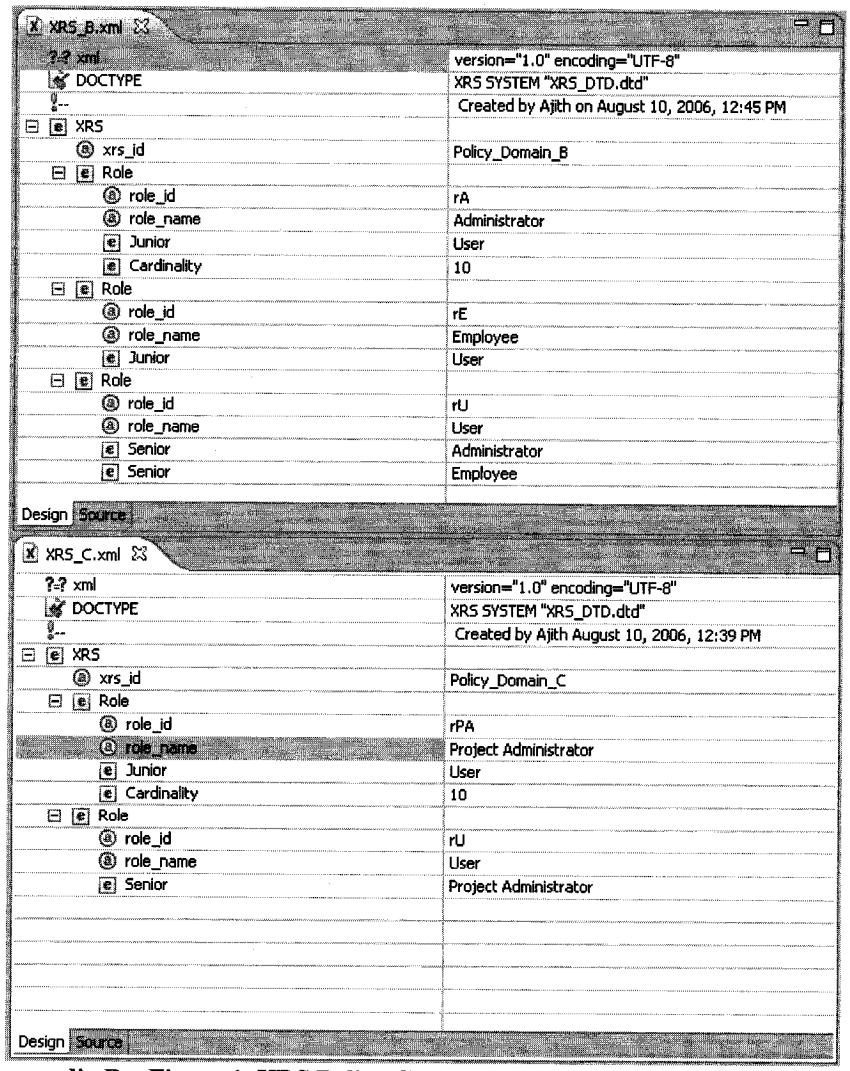
Libraries used:

1. WSS4J version 1.5.0 – a Java library for OASIS Web Service Security, available at http://ws.apache.org/wss4j/
2. Xstream version 1.1.3 – a Java library to serialize objects to XML and back again, available at http://xstream.codehaus.org/
3. JADT – Dictionary and Thesaurus API for Java, available at http://www.alphaworks.ibm.com/tech/jadt

Appendix B:

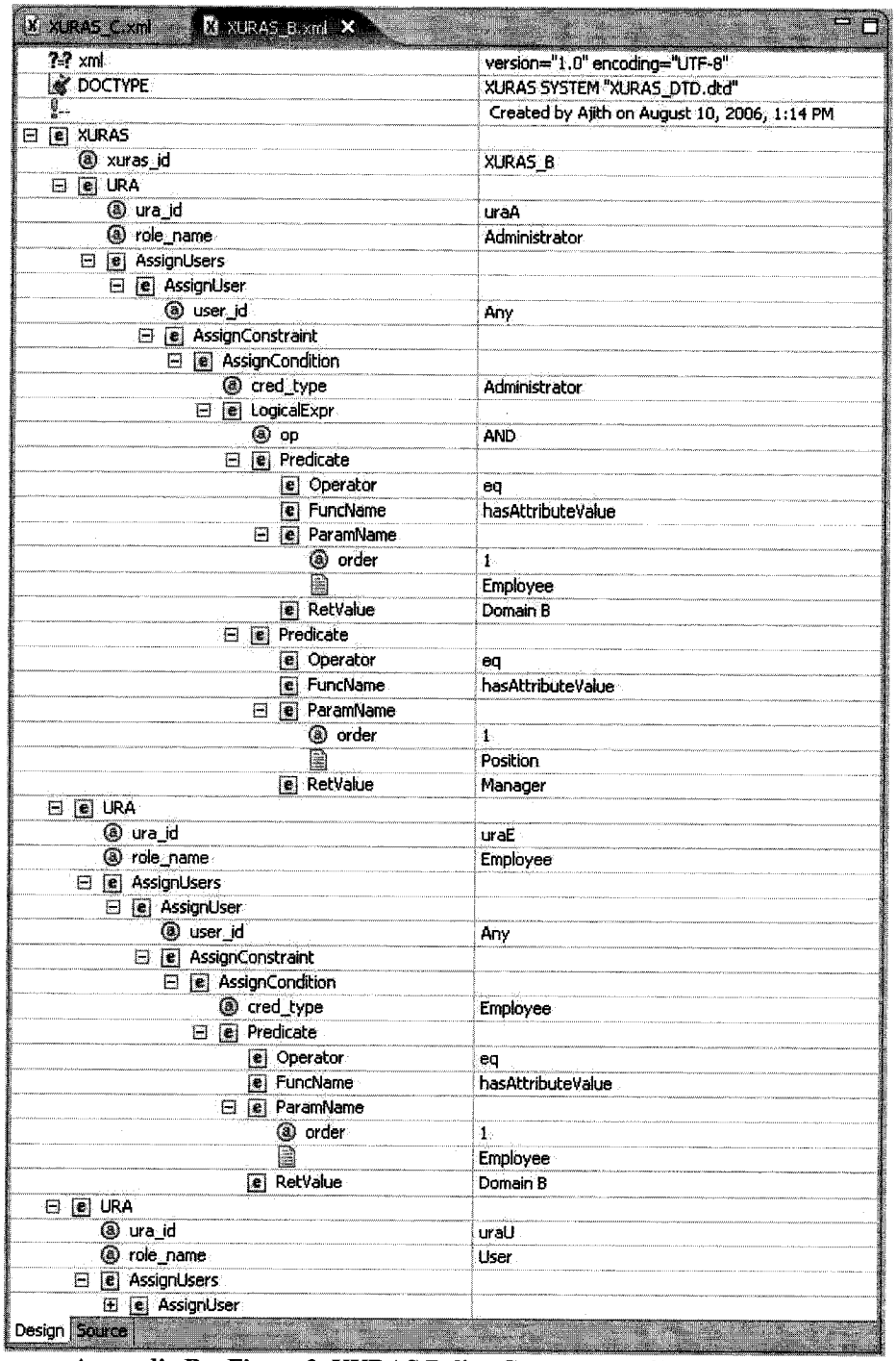
Following are the screen shots of the case study implementation described in Chapter 6:

1. Screen shot of the XRS policy components of Domain B and Domain C, written using Eclipse IDE and an XRS schema.



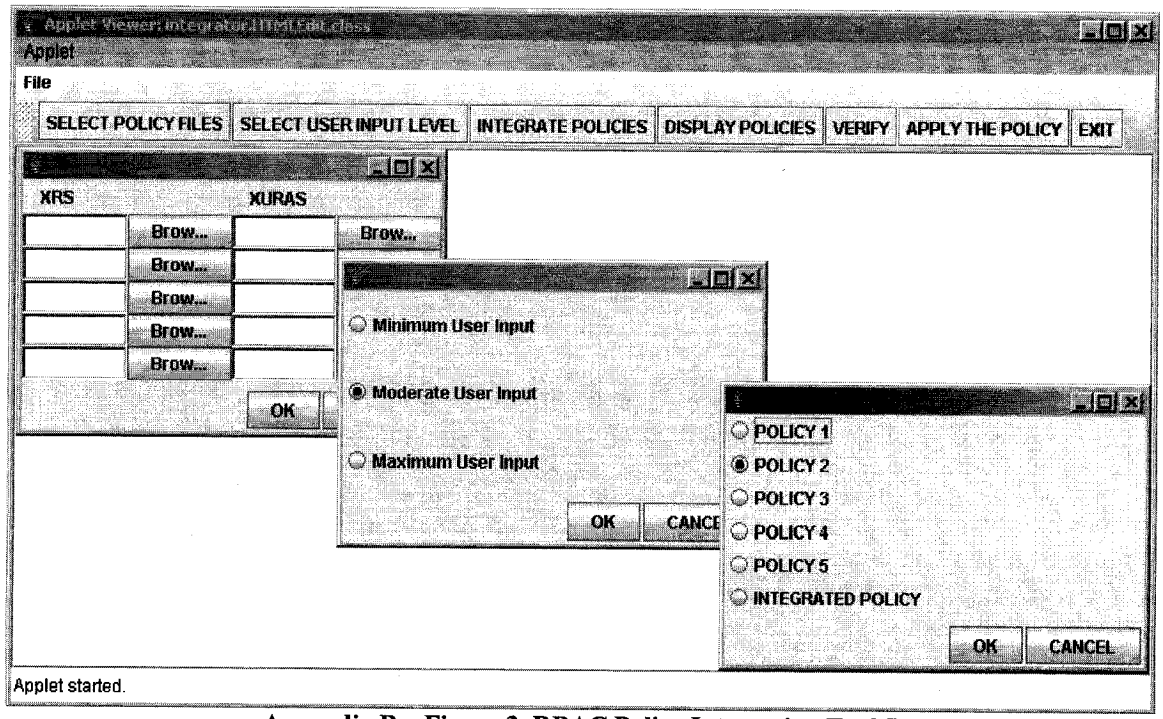
Appendix B – Figure 1. XRS Policy Components of Domain B and Domain C

2. Screen shot of XURAS policy components of Domain B in Eclipse IDE using XURAS schema file.



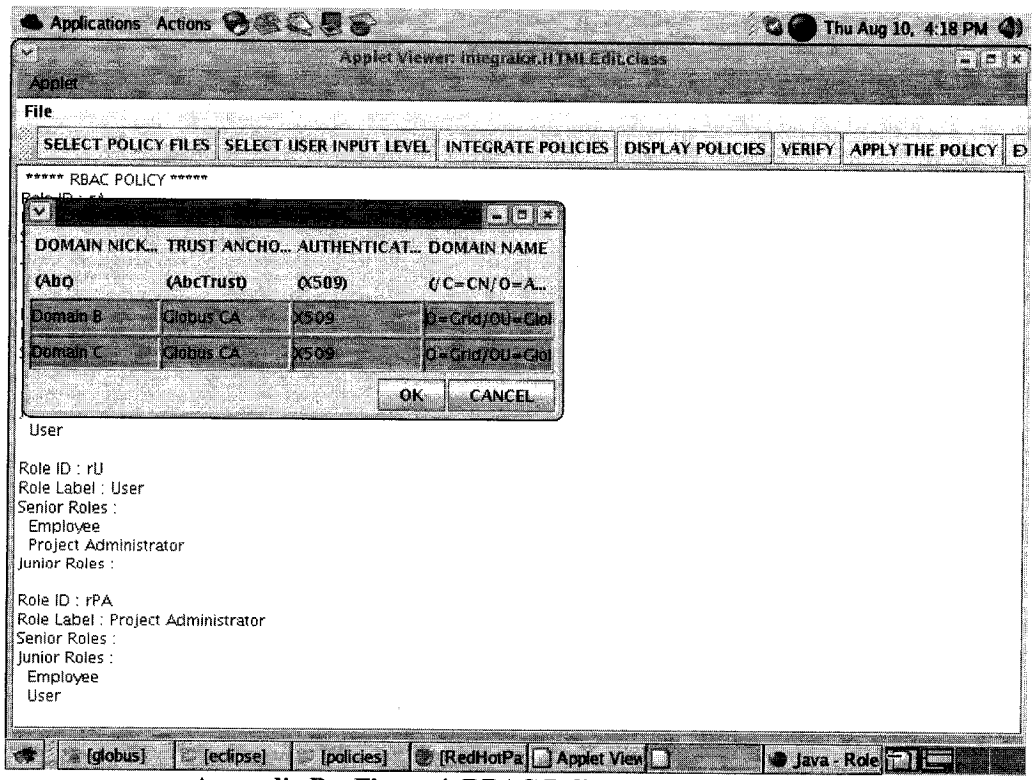
Appendix B – Figure 2. XURAS Policy Component of Domain B

3. The screen shot of the RBAC policy integration tool developed in this thesis. The below figure shows the windows for the domain policy selection, administrator input level, and selected policy display.



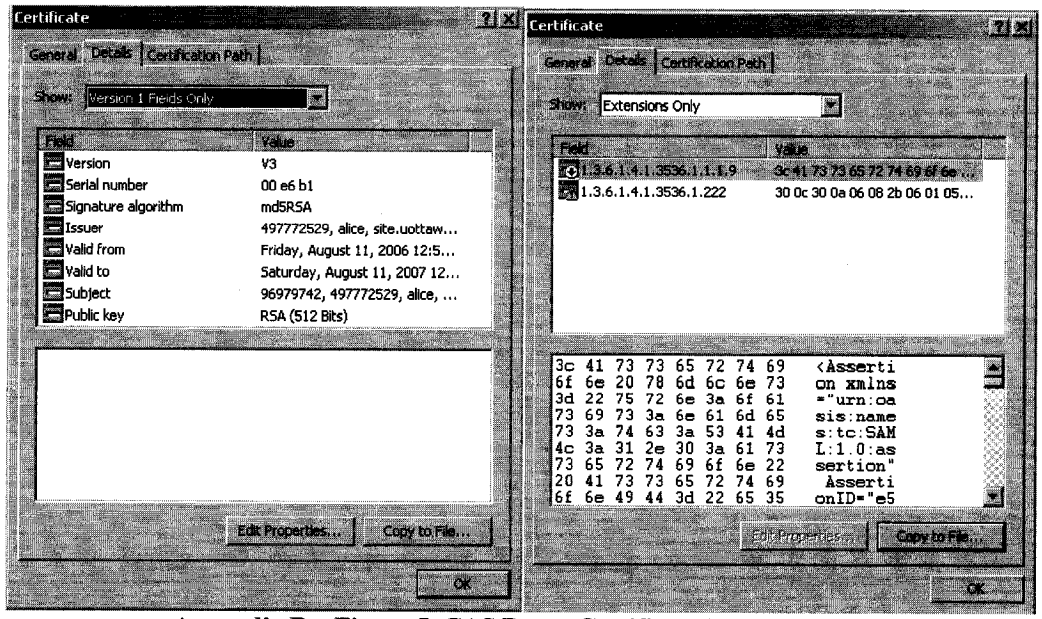
Appendix B – Figure 3. RBAC Policy Integration Tool-I

4. Screen shot of the policy integration tool while specifying the Certification Authority (or Trust Anchor) for each of the domain in the scenario (Domain B and Domain C) before applying the integrated policy into a CAS server.



Appendix B – Figure 4. RBAC Policy Integration Tool-II

5. The CAS proxy certificate issued by the CAS server to *Alice*, member of the Administrator role, is given below. Note: The extension to the certificate includes SAML authorization assertions.



Appendix B – Figure 5. CAS Proxy Certificate Issued to *Alice*

6. The SAML assertions issued to *Alice* are as shown below (*Alice*'s membership to the Administrator role in the community is shown in bold):

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" AssertionID="e506b8a0-295a-11db-
922d-c7b5b9b031e0" IssueInstant="2006-08-11T17:00:30Z"
Issuer="O=Grid,OU=GlobusTest,OU=simpleCA-mcrlab-b50611.site.uottawa.ca,CN=Globus Simple
CA" MajorVersion="1" MinorVersion="0">
    <Conditions NotBefore="2006-08-11T17:00:30Z" NotOnOrAfter="2006-08-12T17:00:30Z">
    </Conditions>
    <AuthorizationDecisionStatement Decision="Permit" Resource="community">
        <Subject>
            <NameIdentifier Format="#X509SubjectName"
NameQualifier="O=Grid,OU=GlobusTest,OU=simpleCA-mcrlab-
b50611.site.uottawa.ca,OU=site.uottawa.ca,CN=alice">/O=Grid/OU=GlobusTest/OU=simpleCA-
mcrlab-b50611.site.uottawa.ca/OU=site.uottawa.ca/CN=alice
            </NameIdentifier>
            <SubjectConfirmation>
                <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:am:X509-PKI
                </ConfirmationMethod>
            </SubjectConfirmation>
        </Subject>
        <Action Namespace="IRBAC">Administrator</Action>
    </AuthorizationDecisionStatement>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">.
        <ds:SignedInfo>.
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315">
            </ds:CanonicalizationMethod>.
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">
            </ds:SignatureMethod>.
            <ds:Reference URI="">.
                <ds:Transforms xmlns:signs="urn:oasis:names:tc:SAML:1.0:assertion">.
                    <ds:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">.
```

```
                        <dsig-xpath:XPath xmlns:dsig-
path="http://www.w3.org/2002/06/xmldsig-filter2"
Filter="intersect">here()/ancestor::signs:Assertion[2]
                        </dsig-xpath:XPath>.
                        <dsig-xpath:XPath xmlns:dsig-
xpath="http://www.w3.org/2002/06/xmldsig-filter2"
Filter="subtract">here()/ancestor::ds:Signature[2]
                        </dsig-xpath:XPath>.
                    </ds:Transform>.
                    <ds:Transform lgorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-
exc-c14n#" PrefixList="code ds kind rw saml samlp signs #default xsd xsi">
                        </ec:InclusiveNamespaces>
                    </ds:Transform>.
                </ds:Transforms>.
                <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>.<ds:DigestValue>i0P
usOtmxDxFEft2SgbCPpObwOc=
                    </ds:DigestValue>.
            </ds:Reference>.
        </ds:SignedInfo>.
        <ds:SignatureValue>.
Li3RsrBHLjmM/NxmJ3eXymfTV1Y2JHOZob6z+J3ELDjB6g7qhkO6ZgLwvAkta+ADll3OPlCO2I6g.mrZ4a2EfEhB/
fFticIO6S0BmSxhURrB+F8bxu1e0tEQ3/RYZDssBxBFkO/iMVehKn//PkpBaHIV.4LV2bFQ+cbH6VJNuJhs=.
        </ds:SignatureValue>.
        <ds:KeyInfo>.
            <ds:X509Data>.
                <ds:X509Certificate>.
MIICbTCCAdagAwIBAgIBATANBgkqhkiG9w0BAQQFADBwMQ0wCwYDVQQKEwRHcmlkMRMwEQYDVQQL.EwpHbG9idXNU
ZXN0MS8wLQYDVQQLEyZzaW1wbGVDQS1tY3JsYWItYjUwNwxLnNpdGUudW90dGF3.YS5jYTEZMBcGA1UEAxMQR2xvY
nVzIFNpbXBsZSBDQTAeFw0wNjA0MDExOTU1MjBaFw0wNzA0MDEx.OTU1MjBaMHIxDTALBgNVBAoTBEdyaWQxEzARB
gNVBAsTCkdsb2J1c1Rlc3QLzAtBgNVBAsTJnNp.bXBsZUNBLW1jcmxhYi1iNTA2bDEuc2l0ZS51b3R0YXdhLmNhMR
swGQYDVQQDExJob3N0L21jcmxh.Yi1iNTA2bDEwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAL4AwcAj/1IvNq
rkta81i/lqQxx.lw2d37QAPXfcoV4qTdh2hI3zSPRkNkTR9UkZ+tqDlqZo+v2wFsMtfzX37+osJgcpBzOiuZGDo4v
j.vJFt3jLoUnh1qvGllv/NMhaqD57OwwPWccOx+ahghyn53IvHSL0CMI10tM/mfmjTrKC9AgMBAAGj.FTATMBEGCW
CGSAGG+EIBAQQEAwIE8DANBgkqhkiG9w0BAQQFAAOBgQDDmQGoyQY87k+6ORpxUId.XcMngcv9gaULAUUgixa52e+
gNLiagmZMeqUc96GmfA3iQXw3OEtouMY+Or+j3IMQYuh4ZYJxKTkL.riVV0wUkA5CUlrhdew6LbaiVJbBOHsK6D0X
Xf6x/iokb/CWXIXqgimT5rRFFDEpqpbJzX6cgQ==.
                </ds:X509Certificate>.
            </ds:X509Data>.
        </ds:KeyInfo>
    </ds:Signature>
</Assertion>
```

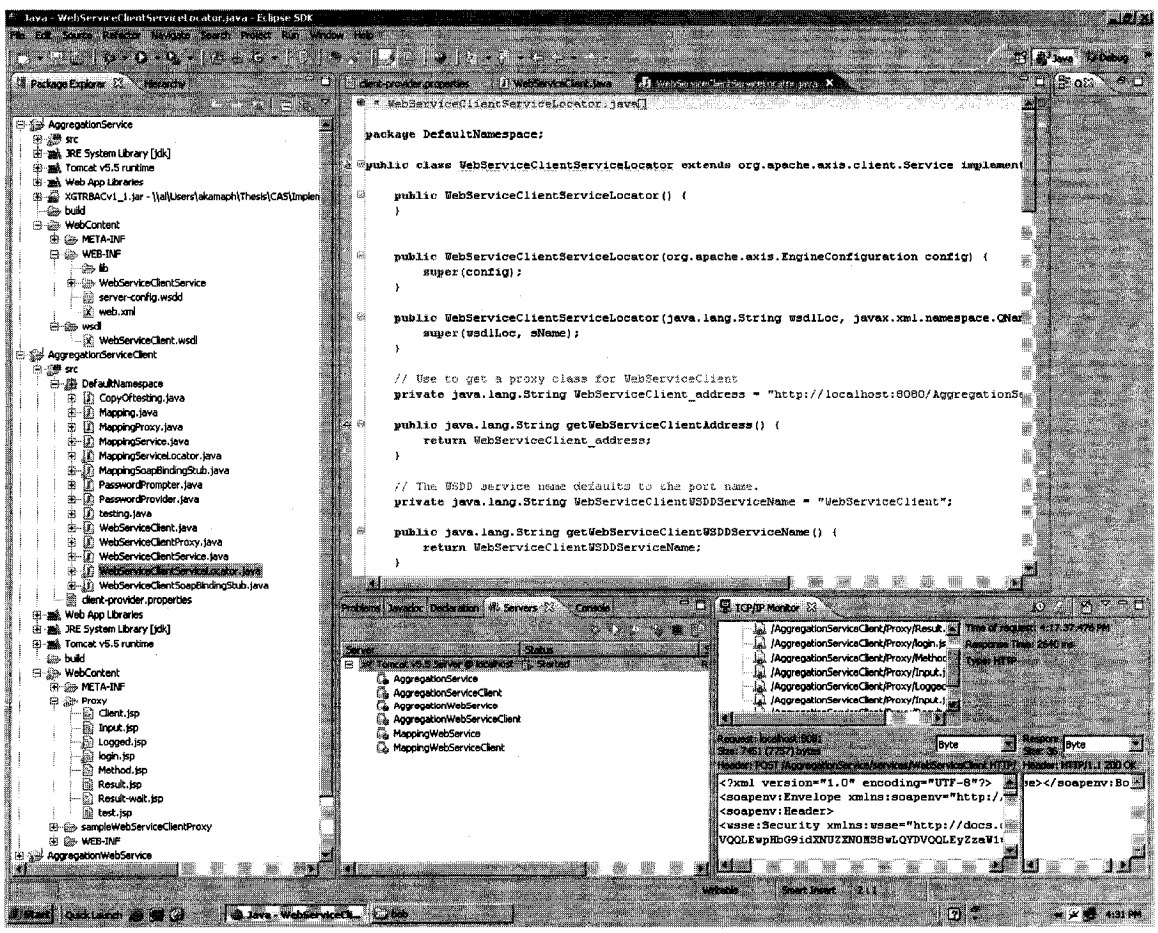
7. The following is the SOAP request message sent by *Alice* to the aggregation web service to invoke the average temperature method. Notice that *Alice* has embedded her CAS proxy certificate in the SOAP header.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header>
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3" wsu:Id="CertId-633096391">
MIIPbTCCDxegAwIBAgIDAOaxMA0GCSqGSIb3DQEBBAUAMIGTMQ0wCwYDVQQKEwRHcmlkMRMwEQYD
VQQLEwpHbG9idXNUZXN0MS8wLQYDVQQLEyZzaW1wbGVDQS1tY3JsYWItYjUwNmwxLnNpdGUudW90
dGF3YS5jYTEYMBYGA1UECxMPc2l0ZS51b3R0YXdhLmNhMQ4wDAYDVQQDEwVhbGljZTESMBAGA1UE
AxMJNDk3NzcyNTI5MB4XDTA2MDgxMTE2NTUzMVoXDTA3MDgxMTE2MjI0NlowgaYxDTALBgNVBAoT
BEdyaWQxEzARBgNVBAsTCkdsb2J1c1Rlc3QxLzAtBgNVBAsTJnNpbXBsZUNBLW1jcmxhYi1iNTA2
bDEuc2l0ZS51b3R0YXdhLmNhMRgwFgYDVQQLEw9zaXRlLnVvdHRhd2EuY2ExDjAMBgNVBAMTBWFs
aWNlMRIwEAYDVQQDEwk0OTc3NzI1MjkxETAPBgNVBAMTCDk2OTc5NzQyMFwwDQYJKoZIhvcNAQEB
BQADSwAwSAJBAK55lvgbVLL7Oofy1B+lY+JezaL7lTiOGKlHj1R0oOZxoZRzo+WDDyXW2WbwQf27
qTxr8fuU61tDPzpEsETlltsCAwEAAaOCDT0wgg05MB8GCisGAQQBm1ABgV4BAf8EDjAMMAoGCCsG
AQUFBxUBMIINFAYLKwYBBAGbUAEBAQkEgg0DPEFzc2VydGlvbiB4bWxucz0idXJuOm9hc2lzOm5h
```

bWVzOnRjOlNBTUw6MS4wOmFzc2VydGlvbiIgQXNzZXJ0aW9uSUQ9ImU1MDZiOGEwLTI5NWEtMTFk
Yi05MjJkLWM3YjViOWIwMzFlMCIgSXNzdWVJbnN0YW50PSIyMDA2LTA4LTExVDE3OjAwOjMwWilg
SXNzdWVyPSJPPUdyaWQsT1U9R2xvYnVzVGVzdCxPVT1zaW1wbGVDQS1tY3JsYWltYjUwNmwxLnNp
dGUudW90dGF3YS5jYSxDTj1HbG9idXMgU2ltcGxlIENBIiBNYWpvclZlcnNpb249IjEiIE1pbm9y
VmVyc2lvbj0iMCI+PENvbmRpdGlvbnMgTm90QmVmb3JlPSIyMDA2LTA4LTExVDE3OjAwOjMwWilg
Tm90T25PckFmdGVyPSIyMDA2LTA4LTEyVDE3OjAwOjMwWil+PC9Db25kaXRpb25zPjxBdXRob3Jp
emF0aW9uRGVjaXNpb25TdGF0ZW1lbnQgRGVjaXNpb249IlBlcm1pdCIgUmVzb3VyY2U9IkZUUERp
cmVjdG9yeVRyZWV8Y29tbXVuaXR5Ij48U3ViamVjdD48TmFtZUlkZW50aWZpZXIgRm9ybWF0PSIj
WDUwOVN1YmplY3ROYW1lIiBOYW1lUXVhbGlmaWVyPSJPPUdyaWQsT1U9R2xvYnVzVGVzdCxPVT1z
aW1wbGVDQS1tY3JsYWltYjUwNmwxLnNpdGUudW90dGF3YS5jYSxPVT1zaXRlLnVvdHRhd2EuY2Es
Q049YWxpY2UiPi9PPUdyaWQvT1U9R2xvYnVzVGVzdC9PVT1zaW1wbGVDQS1tY3JsYWltYjUwNmwx
LnNpdGUudW90dGF3YS5jYS9PVT1zaXRlLnVvdHRhd2EuY2EvQ049YWxpY2U8L05hbWVJZGVudGlm
aWVyPjxTdWJqZWN0Q29uZmlybWF0aW9uPjxDb25maaXJtYXRpb25NZXRob2Q+dXJuOm9hc2lzOm5h
bWVzOnRjOlNBTUw6MS4wOmFtOlg1MDktUEtJPC9Db25maaXJtYXRpb25NZXRob2Q+PC9TdWJqZWN0
Q29uZmlybWF0aW9uPjwvU3ViamVjdD48QWN0aW9uIE5hbWVzcGFjZT0iSVJCQUMiPkFkbWluaXN0
cmF0b3I8L0FjdGlvbj48L1dhcml6YXRpb25EZWNpc2lvbIN0YXRlbWVudD48ZHM6U2lnbmF0
dXJlIHhtbG5zOmRzPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjIj4KPGRzOlNp
Z25lZEluZm8+CjxkczpDYW5vbmljYWxpemF0aW9uTWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3
dy53My5vcmcvMjAwMS9SRUMteG1sLWMxNG4tMjAwMTAzMTUiPjwvZHM6Q2Fub25pY2FsaXph
dGlvbk1ldGhvZD4KPGRzOlNpZ25hdHVyZU1ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMu
b3JnLzIwMDAvMDkveG1sZHNpZyNyc2Etc2hhMSI+PC9kczpTaWduYXR1cmVNZXRob2Q+CjxkczpS
ZWZlcmVuY2UgVVJJPSIiPgo8ZHM6VHJhbnNmb3JtcyB4bWxuczpzaWducz0idXJuOm9hc2lzOm5h
bWVzOnRjOlNBTUw6MS4wOmFzc2VydGlvbiI+CjxkczpUcmFuc2Zvcm0gQWxnb3JpdGhtPSJodHRw
Oi8vd3d3LnczLm9yZy8yMDAyLzA2L3htbC1leGMtYzE0biMiPjwvZHM6VHJhbnNmb3JtPgo8ZHM6
IHhtbG5zOmRzaWcteHBhdGg9Imh0dHA6Ly93d3cudzMub3JnLzIwMDIvMDYveG1sZHNpZy1maWx0
ZXIyIiBGaWx0ZXI2VjdCI+aGVyZSgpL2FuY2VzdG9yOjpzaWducz0idXJuOm9hc2lzOm5h
bWVzOnRjOlNBTUw6MS4wOmFzc2VydGlvbiI+CjxkczpUcmFuc2Zvcm0gQWxnb3JpdGhtPSJodHRw
Oi8vd3d3LnczLm9yZy8yMDAyLzA2L3htbC1leGMtYzE0biMiPjwvZHM6VHJhbnNmb3JtPgo8ZHM6
IHhtbG5zOmRzaWcteHBhdGg9Imh0dHA6Ly93d3cudzMub3JnLzIwMDIvMDYveG1sZHNpZy1maWx0
ZXIyIiBGaWx0ZXI9ImludGVyc2VjdCI+aGVyZSgpL2FuY2VzdG9yOjpzaWducz0idXJuOm9hc2lz
bWVzOnRjOlNBTUw6MS4wOmFzc2VydGlvbiI+CjxkczpUcmFuc2Zvcm0gQWxnb3JpdGhtPSJodHRw
Oi8vd3d3LnczLm9yZy8yMDAyLzA2L3htbC1leGMtYzE0biMiPjwvZHM6VHJhbnNmb3JtPgo8ZHM6
YTEiPjwvZHM6RGlnZXN0TWV0aG9kPgo8ZHM6RGlnZXN0VmFsdWU+aTBQdXNPdG14RHhGRWZ0MlNn
YkNQcE9id09jPTwvZHM6RGlnZXN0VmFsdWU+CjwvZHM6UmVmZXJlbmNlPgo8ZHM6UmVmZXJlbmNlIFBgo8ZHM6RGln
Zm8+CjxkczpTaWduYXR1cmVWYWx1ZT4KTGkzUnNyQkhMam1NL054bUozZVh5bWZUVjFZMkplT1pv
YjZ6K0ozRUxEakI2ZzdxaGtPNlpnTHd2QWt0YStBRGxsM09QbENPMkk2Zwptclo0YTJFZkVoQi9m
RnRpY0lPNlMwQm1TeGhVUnJCK0Y4Ynh1MWUwdEVRMy9SWVpEc3NCeEJGa08vSWlNVmVoS24vL1Br
cEJhSElWCjRMVjJiRlErY2JlNlZKTnVKaHM9CjwvZHM6U2lnbmF0dXJlVmFsdWU+CjxkczpzIZ
bmZvPgo8ZHM6WDUwOURhdGE+CjxkczpYNTA5Q2VydGlmaWNhdGU+Ck1JSUNiVENDQWRhZ0F3SUJB
Z0lCQVRBTkJna3Foa2lHOXcwQkFRUUZBREJ3TVEwd0N3WURWUVFLRXdSSGNtbGtNUk13RVFZRFZR
UUwKRXdwSGJHOWlkWE5VWlhOME1TOHdMUVlEVlFRTEV5WnphVzF3YkdWRFFTMXRZM0pzWVdJdFlq
VXdObXd4TG5OcGRHVXVkVzkwZEdGMwpZUzVqWVRFWk1CY0dBMVVFQXhNUVlyeHZZblZ6SUZOcGJY
QnNaU0JEUVRBZUZ3MHdOakEwTURFeE9UVTFNakJhRncwd056QTBNREV4Ck9UVTFNakJhTUhJeERU
QUxCZ05WQkFvVEJFZHlhV1F4RXpBUkJnTlZCQXNUQ2tkc2lySjFjMVJsYzNReEx6QXRCZ05WQkFz
VEpuTnAKYlhCc1pVTkJMVzFqYTJ4aFlpMWlOVEEyYkRFdWMybDBaUzUxYjNSMFlYZGhMbU5oTVJz
d0dRWURWUVFERXhKb2IzTjBMMjFqY214aApZaTFpTlRBMmJERXdnWjh3RFFZSktvWklodmNOQVFF
QkJRQURnWTBBTUlHSkFvR0JBTDRBd2NBai8xSXZOcXFya3RhODFpL2xxUXh4Cmx3MmQzN1FBUFhm
Y29WNHFUZGgyaEkzelNQUmtOa1RSOVVrWit0cURscVpvK3Yyd0ZzTXRmelgzNytvc0pnY3BCek9p
dVpHRG80dmoKdkpGdDNqTG9VbmgxcXZHbGx2L05NaGFxRDU3T3d3UFdjY094K2FoZ2h5bjUzSXZl
U0wwQ01JMTB0TS9tZm1qVHJLQzlBZ01CQUFHagpGVEFUTUJFR0NXQ0dTQUdHK0VJQkFRUUVBd0lF
OERBTkJna3Foa2lHOXcwQkFRUUZBQU9CZ1FERG1RckdveVFZODdrKzZPUnB4VUlkClhjTW5nY3Y5
Z2FVTEFVWWdpeGE1MmUrZ05MaWFnbVpNZXFVYzk2R21mQTNpUVh3M09FdG91TVkrT3lrajNJTVFZ
dWg0WlIKeEtUa0wKcmlWVjB3VWtBNUNVbHJoZGV3NkxiYWlWSmJCT0hzSzZEMFhYZjZ4L2lva2lv
Q1dYSVhxZ2ltVDVyUkZGREVwcXBiSnpYVTZjZ1E9PQo8L2RzOlg1MDIDZXJ0aWZpY2F0ZT4KPC9k
czpYNTA5RGF0YT4KPC9kczpLZXlJbmZvPjwvZHM6U2lnbmF0dXJlPjwvZXM6U2lnW9uPjANBgkq
hkiG9w0BAQQFAANBADFj28l6Mqk8BXDzh2X5laQd5XKXqPZlq8/ilKlnk5rvevB1MyPWY4+G8uag
UOaPwxVUu9f+CEJbtELKluLxaT4=
</wsse:BinarySecurityToken>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></ds:SignatureMethod>
<ds:Reference URI="#id-6326112">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>

```
      <ds:DigestValue>jLy0iOOad0o65nriiWtFwXnzyXo=</ds:DigestValue>
    </ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
CT+70iQyPBrLNWB2uH5l4/LF3nbLSCj1BELYycApWe7NPFWr0FOYNyuNfa+73P0ugPCrumBw6Df7
KxyiE6KNYQ==
</ds:SignatureValue>
<ds:KeyInfo Id="KeyId-26977856">
   <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd" wsu:Id="STRId-29876954">
      <wsse:Reference URI="#CertId-633096391" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3">
      </wsse:Reference>
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>
</soapenv:Header>
<soapenv:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="id-6326112">
   <tempAvg xmlns="http://DefaultNamespace">
   </tempAvg>
</soapenv:Body>
</soapenv:Envelope>
```

8. The Aggregation web service client program in Eclipse IDE 3.2.1 (showing the TCP/IP monitor and Tomcat server status windows):



Appendix B – Figure 6. Eclipse IDE for Aggregation Web Service Client Project